华恒科技

HHARMS3C6410产品技术手册



飞思卡尔半导体全球设计联盟成员



美国ADI公司DSP技术合作伙伴



ARM Connected Community

安 全 警 示

首先感谢您选择华恒科技的产品，为了保证安全，请不要在带电的情况下进行以下操作：
1.拔插核心板或内存条，核心板拔下后重新插上后要确保 USB 接口朝外
2.拔插串口线
3.拔插 JTAG/BDM
4.拔插存储卡(SD/CF)
5.拔插 PCI 卡
6.拔插 IDE 设备

另外请注意：
1.静电防护
2.勿在临界状态下工作
3.勿超过产品最大承受值

如果您在以后使用华恒产品时，遇到故障，请阅读本手册最后的《售后服务与技术支持》一节，寻求华恒科技的帮助。

版 本 声 明

本手册适用于 HHARM64xx 系列平台(HHARM6410,HHARM6410)，最近更新的手册一般情况下适用于以前的同一种型号 CPU 的其它平台。

本手册的说明主要是针对华恒提供的 HHARM6410 系列产品，手册中难免存在一些错误及不足，敬请客户谅解，并真诚地欢迎您提出宝贵的意见和建议，华恒科技会定期作出修改。

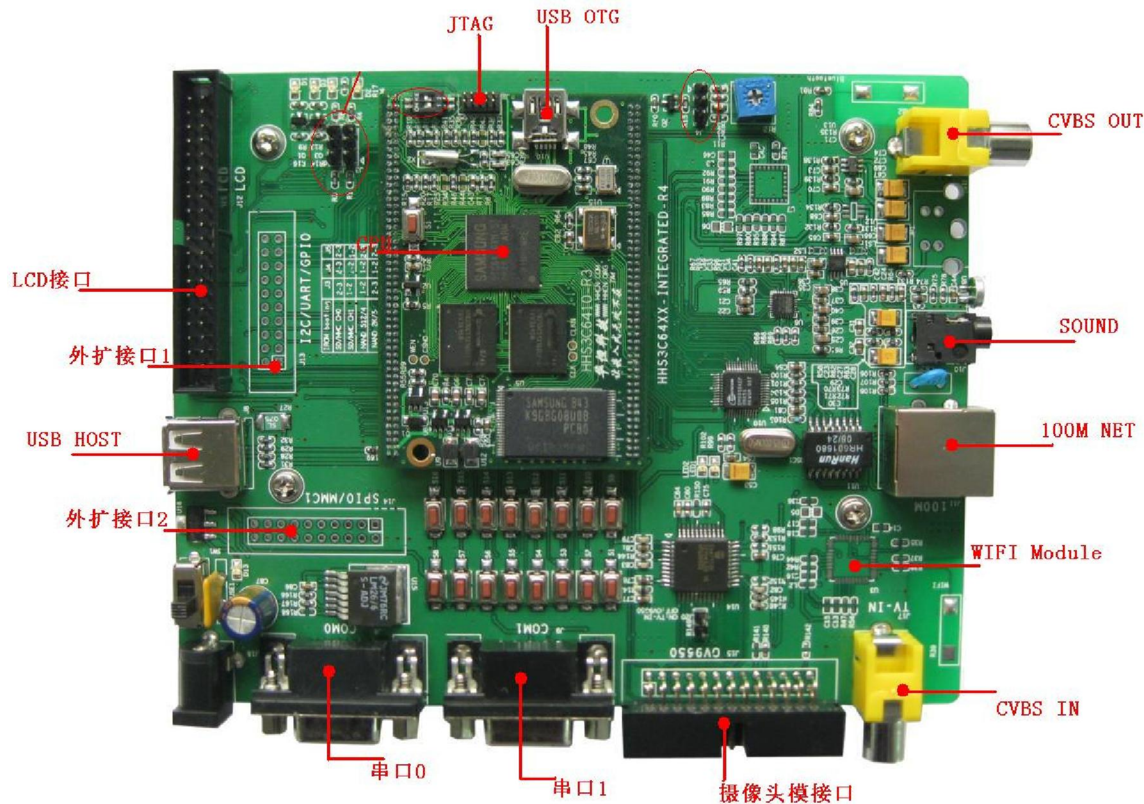
修 订 描 述			
日期	修订版本	描述	编辑
……	……	……	……
2008.8	V1.0	初稿	梁振成
2008.8	V1.1	修订和完善	梁振成 陈宜义 鲍小路
2008.8	V1.5	完善	梁振成 陈宜义 鲍小路
2008.8.26	V1.8	增加 ubuntu 中的 tftp 及 nfs 及其它配置，修改手册中的称谓	王 康
2008.9.12	V1.9	修改附录 F，使用仿真器烧写 nand部分	杨贤良
2009-3-23	V2.0	针对新板修改部分说明	杨贤良
2009-4-16	V2.1	添加 wi fi 的测试说明，部分参数相关测试程序及 QT	陈春生

目 录

第一章 快速上手指南

在本章中您可以对 HHARM6410 开发板有个初步的认识，对开发板进行简单检测及建立开发环境。

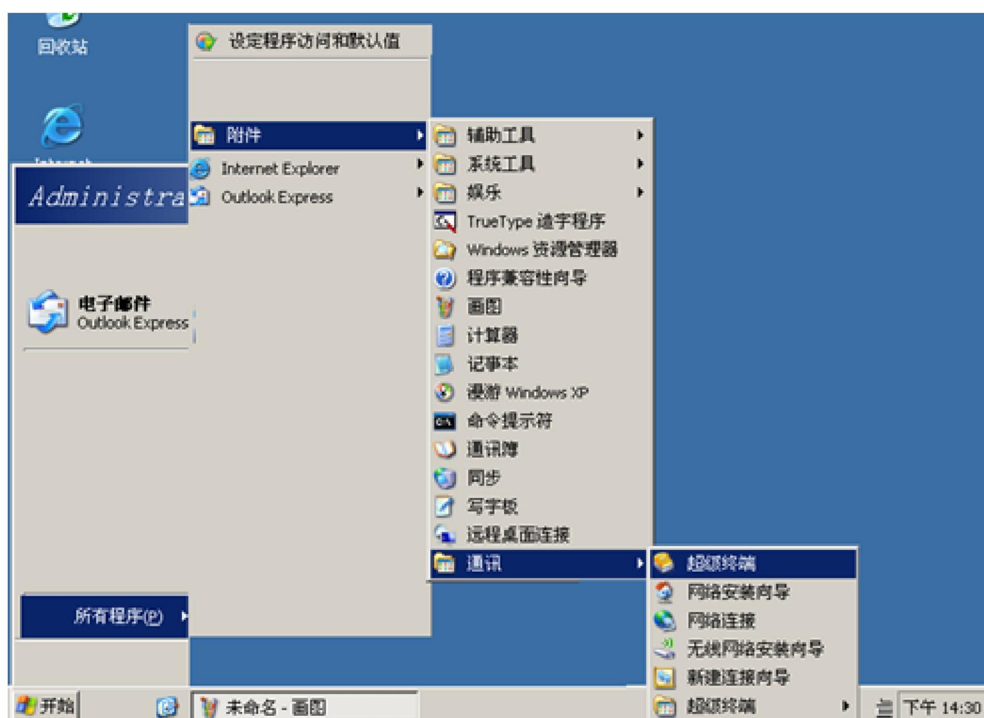
1.开发板的简单测试



（图 1 HHARM6410 开发板实物图）

在本章节中您可以对 HHARM6410 开发套件有个初步的认识，对开发板进行简单检测及建立开发环境。Windows 98、2000、XP 安装盘中都附有超级终端应用程序。如果您的系统没有安装超级终端，您可以在控制面板“添加/删除程序”一栏选择安装超级终端应用程序。

首先在 Windows XP 环境下从开始菜单启动超级终端。



启动后如下图：



请输入连接名称，这个名称可以为任意字符，比如输入“HHTECH”，然后点击“确定”按钮。



设置连接使用 COM1（串口 1），点击“确定” 按钮进入 COM1 的属性设置窗口，如下图。



请按照上图正确设置串口(每秒位数为 115200,数据位 8,奇偶校验无,停止位 1,数据流控“无”),确定后进入超级终端窗口。

请您使用产品附带的串口对接线将开发板串口 0（见图）与 PC 机的串口 1 连接。接通开发板 9V/12V 电源后超级终端中将会显示开发板的启动信息，如下图：

```
115200 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
asoc: WM8987 <-> s3c-i2s mapping ok###################
WM8987: 0x33c2, R25 <= 0x1c2
block
WM8987: 0x4550, R34 <= 0x150

WM8987: 0x2de0, R22 <= 0x1e0
WM8987: 0x15c7, R10 <= 0x1c7
WM8987: 0x17c7, R11 <= 0x1c7
WM8987: 0x0177, R00 <= 0x177
WM8987: 0x0357, R01 <= 0x157
WM8987: 0x5079, R40 <= 0x079
WM8987: 0x537a, R41 <= 0x17a
init WM8987 OK
Proc-FS interface for audio codec
ALSA device list:
  #0: smdk6400 (WM8987)
TCP cubic registered
s3c2410-rtc s3c2410-rtc: hctosys: invalid date/time
Freeing init memory: 5124K
init started: BusyBox v1.12.0 (2008-07-31 14:42:10 HKT)
starting pid 246, tty '': '/etc/init.d/rcS'
eth0: link down
WM8987: 0x3343, R25 <= 0x143
eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
NET: Registered protocol family 1

Please press Enter to activate this console.
starting pid 261, tty '': '-/bin/sh'
# _
已连接 0:08:40 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

需要按一下 Enter(回车键)，即可进入 shell 提示符“ # ”。现在就可以键入 Linux 的命令操作开发板了，例如可以键入 ls 命令列出开发板已经存在哪些目录,如下图。

```
115200 - 超级终端
文件(F) 编辑(E) 查看(V) 呼叫(C) 传送(T) 帮助(H)
WM8987: 0x4550, R34 <= 0x150
WM8987: 0x4b50, R37 <= 0x150
done han
Bytes

WM8987: 0x0177, R00 <= 0x177
WM8987: 0x0357, R01 <= 0x157
WM8987: 0x5079, R40 <= 0x079
WM8987: 0x537a, R41 <= 0x17a
init WM8987 OK
Proc-FS interface for audio codec
ALSA device list:
  #0: smdk6400 (WM8987)
TCP cubic registered
s3c2410-rtc s3c2410-rtc: hctosys: invalid date/time
Freeing init memory: 5124K
init started: BusyBox v1.12.0 (2008-07-31 14:42:10 HKT)
starting pid 246, tty '': '/etc/init.d/rcS'
eth0: link down
WM8987: 0x3343, R25 <= 0x143
eth0: link up, 100Mbps, full-duplex, lpa 0x45E1
NET: Registered protocol family 1

Please press Enter to activate this console.
starting pid 261, tty '': '-/bin/sh'
# ls
bin       hhtech    sbin      var       init      mnt       etc
sys       dev       usr       proc      lib       tmp       linuxrc
# _
已连接 0:11:14 ANSIW 115200 8-N-1 SCROLL CAPS NUM 捕 打印
```

若按以上操作步骤启动开发板时工作不正常,请立即拨打电话与华恒公司联系(联系方式请见本手册的最后一页)，华恒科技将为您提供指导意见或更换开发板。您也可以到华恒科技网站上去看一看 FAQ(常见问题解答)。华恒 HHARM 系列嵌入式 LINUX 开发套件常见问题解答最新版本请见 www.hhcn.com/chinese/hharmfaq.html。

在 Linux 操作系统下，用户可以使用 minicom 这个程序来连接开发板，具体设置请参照超级终端。

第二章 产 品 简 介

HHARM6410 是华恒科技推出的用于高端手持设备、微型智能控制设备的开发套件。采用韩国三星公司的 ARM11 内核的处理器 S3C6410/S3C6410。该款套件核心板的尺寸仅相当于一个 48mm*67mm 的方块的大小。

HHARM6410 套件由核心板和底板（外设板或称基本板）组成，核心板上集成三星 S3C6410 处理器，128MB 的 DDR 内存以及 1GB 的 NANDFLASH，同时预留了 256K NORFLASH。为您的应用研发提供了充足的空间。底板上则提供以下外设接口：

1. 两个四线 RS-232 串口(COM0，COM1)
2. 一个 USB HOST 接口（USB device 接口在核心板上）
3. 一个 10M/100M 自适应以太网接口，一个 TFT LCD 接口,一个触摸屏接口
4. 一个 wm8987 sound 接口
5. 一个 4x4 按键接口
6. 一路视频输入（模拟 saa7113 或数字 ov9650，可以选择）
7. 一路视频输出 TVOUT
8. 一个 RTC 和 watchdog
9. 一个 SDIO 接口 WIFI 模块
10. 一个 SD 卡接口

核心板和底板配合即构成一个最小的完整应用系统。系统具有体积小、耗电低、处理能力强等特点，能够装载和运行嵌入式 Linux 操作系统。用户可以在这个系统平台上进行自主软件开发。

HHARM6410 套件中提供底板硬件电路图和硬件设计文档，极大的方便了用户进行硬件扩展开发。同时华恒科技提供完备的嵌入式 Linux 开发环境及丰富的开发调试工具软件。

S3C6410 微处理器的特性：

ARM 11 嵌入式处理器内核，主频可达 667MHz；
扩展总线最大频率 133MHz；
32位数据总线和 32位外部地址总线 ；
完全静态设计 (0-667M)；
存储控制器 (八个存储体)：
包含 SROM、 SRAM 控制器 NAND 控制器 ；
复位时引导芯片选择 (8比特、 16比特存储或 NAND 可供选择)；
五个三十二位定时器，(time0,time1带有 PWM);
多达 64个中断源的中断控制器 ；
RTC；
四个 UART， Supports IrDA 1.0；
四个 DMA控制器,每个 DMA控制器有 8个 通道 ;(支持外设 DMA) 支持 STN与 TFT LCD 控制器；
看门狗；
IIS音频接口；
两个 USB host口 ,一个 USB device口。
IIC-Bus接口（6410 仅一个， 6410支持两个)；
两个串行外围接口电路（SPI)
三个 SD 卡接口（sopport 1/4/8 bitmode,rate up to 50MHz);
8x8自定义按键
.camera_if接口
TV_out接口
MFC(多格式视频编解码)接口，支持 H263、 H264、 MPEG4和 VC-1硬件编解码。

HHARM6410 开发套件硬件主要结构：

Sumsung S3C6410 处理器
1Gbytes 8 位 NAND FLASH
2X64Mbytes 16 位 DDRRAM
两个四线 RS-232 接口
一个 10M/100M 自适应以太网接口
camera摄像头接口 (可选 saa7113或 ov9650)
两个 USB(一个 host,一个 device)接口
一个 SD卡接口
4X4自定义按键
RTC
看门狗
TVOUT

IIS音频（使用 wm8987芯片）
一个 TFT LCD接口 (可选两种不同接口的屏）
一个触摸屏接口。
MFC 接口
9V 直流电源（需要客户自己购买）
H/W 复位建
运行状态指示 LED 灯
WIFI模块实现无线通讯功能

第三章 软 件 系 统

HHARM 6410为一台采用 S3C6410 处理器、提供 RS232 接口，安装有华恒嵌入式 Linux 操作系统的软硬件开发平台，其功用相当于一台装有 Linux，装有串口的 PC 机。对于 HHARM 6410，它提供的所有软件（操作系统和应用软件）都固化在板上 FLASH 里面，就相当于 PC 机的硬盘。FLASH 上的内容，包括 Bootloader都可通过烧写工具来更新升级。用户可以在 HHARM 6410平台上开发应用程序或更改其上的操作系统的工作方式（因为操作系统是开放源代码的），和在 PC 上开发应用唯一的不同之处在于它要采用一种交叉编译的开发模式，即为 HHARM 6410开发驱动及应用时，把相关代码的编辑、编译等工作寄宿到一台 PC 机上去完成，PC 机上编译后的二进制代码想办法放到开发板上去执行的方式，而不是在开发板上去编辑、编译相关程序。详细介绍请参见后面的章节。

随着微处理器的产生，价格低廉、结构小巧的 CPU 和外设连接提供了稳定可靠的硬件架构，那么限制嵌入式系统发展的瓶颈就突出表现在软件方面。尽管从八十年代末开始，陆续出现了一些嵌入式操作系统，比较著名的有 Vxwork、pSOS、Neculeus和 Windows CE。但这些专用操作系统都是商业化产品，其高昂的价格使许多对成本要求比较严格的公司或个人望而却步；而且，源代码的封闭性也大大提高开发难度和限制了开发者的积极性。另外，结合国内实情，当前国家对自主操作系统的大力支持，也为源码开放的 Linux 的推广提供了广阔的发展前景。还有，对上层应用开发者而言，嵌入式系统需要的是一套高度简练、界面友善、质量可靠、应用广泛、易开发、多任务，并且价格低廉的操作系统。我们可以预见，在不久的将来，嵌入式设备会无处不在。

因为 Linux 的开放性， Linux 非常适合多数 Internet 设备。Linux 可以支持不同的设备，支持不同的配置。Linux 对厂商不偏不倚而且成本极低，能够很快成为用于各种设备的操作系统。如今，业界已经达成共识：即嵌入式 Linux 是大势所趋，其巨大的市场潜力与无限商机必然会进一步吸引众多的厂商进入这一领域。

嵌入式操作系统主要有 Palm OS，Windows CE，EPOC，LinuxCE，QNX，ECOS，高端嵌入式系统要求许多高级的功能，如图形用户界面和网络支持。很多高端 RTOS 供应商已经提供了这些功能，但其价格也很高，一般人难以接受。微软的 Windows CE 也有此类功能，却不具备大多数嵌入式系统要求的实时性能，而且难以移植。作为一个便宜、成熟并且提供高端嵌入式系统所必须特性的操作系统，嵌入式 Linux 操作系统以价格低廉、功能强大又易于移植而正在被广泛采用，成为新兴的力量，所以，众多商家纷纷转向了嵌入式 Linux。

Linux为嵌入操作系统提供了一个极有吸引力的选择，它是个和 Unix相似、以核心为基础的、完全内存保护、多用户、多任务的操作系统。支持广泛的计算机硬件 包括 X86，Alpha，Sparc，MIPS，PPC，ARM，等现有的大部分芯片。软件源码全部公开，任何人可以修改并在 GPL协议（GNU General Public License)下发行 这样，开发人员可以对操作系统进行定制，再也不必担心像 MicrosoftWindows 操作系统中"后门"的威胁。同时由于有 GPL 的控制 大家开发的东西又大都相互兼容，不会走向分裂之路。Linux 用户遇到问题时可以通过 Internet 向网上成千上万的 Linux 开发者请教 这使最困难的问题也有办法解决。Linux 带有 Unix 用户熟悉的完善的开发工具，几乎所有的 Unix 系统的应用软件都已移植到了 Linux上。Linux 还提供了强大的网络功能，有多种基于 X Window的窗口管理器可供选择，其强大的 GNU工具链可以很容易得到，不但成熟完善、而且使用方便。

嵌入式系统选择 Linux 的原因：

可应用于多种硬件平台。Linux 已经被移植到多种硬件平台，这对受开销、时间限制的研究与开发项目是很有吸引力的。原型可以在标准平台上开发然后移植到具体的硬件上，加快了软件与硬

件的开发过程。

Linux可以随意地配置不需要任何的许可证或商家的合作关系。它是免费的，源代码可以得到。这是最吸引人的。毫无疑问，这会节省大量的开发费用。

微内核直接提供网络支持，而不必象其他操作系统要外挂 TCP/IP协议包。

Linux的高度模块化使添加部件非常容易。

Linux在台式机上的成功，也保证了 Linux在嵌入式系统中的辉煌前景。

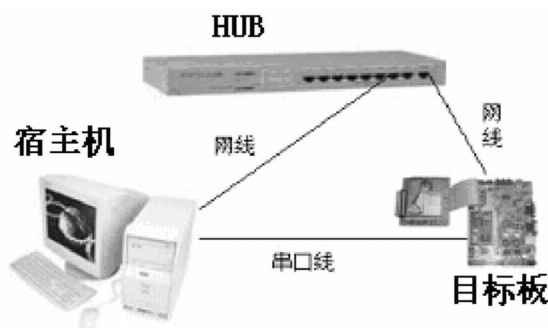
Linux 是一种很受欢迎的操作系统，与 UNIX 系统兼容，开放源代码，广泛应用于服务器领域。而更大的影响在于它正逐渐的应用于嵌入式系统领域。

Linux是一个自由开放的世界，在 Linux（无论 PC 还是嵌入式系统）上进行软件开发都可以在广袤的网络资源中获取帮助。

3.1 使用说明

3.1.1 建立宿主机开发环境

绝大多数的 Linux 软件开发都是以 native 方式进行的，即本机（HOST）开发、调试，本机运行的方式。这种方式通常不适合于嵌入式系统的软件开发，因为对于嵌入式系统的开发，没有足够的资源在本机（即开发板上系统）运行开发工具和调试工具。通常的嵌入式系统的软件开发采用一种叫做交叉编译调试的方式。交叉编译调试环境建立在宿主机（即一台 PC 机）上，对应的开发板叫做目标板。



开发时使用宿主机上的交叉编译、汇编及连接工具形成可执行的二进制代码（这种可执行代码并不能在宿主机上执行，而只能在目标板上执行），然后把可执行文件下载到目标机上运行。调试时的方法很多，可以使用串口、以太网口等，具体使用哪种调试方法可以根据目标机处理器所提供的支持作出选择。宿主机和目标板的处理器一般都不相同，宿主机为 Intel或 AMD处理器， 而目标板如 HHARM 6410 为 SAMSUNG S3C6410处理器，GNU编译器提供这样的功能，在编译编译器时可以选择开发所需的宿主机和目标机从而建立开发环境。所以在进行嵌入式开发前第一步的工作就是要安装一台装有指定操作系统的 PC 机作宿主开发机，宿主机上的操作系统一般要求安装 Linux，但 Linux有多个发行版本，在此，华恒科技推荐用户使用 Ubuntu作为本套开发系统的宿主机 PC操作系统。然后在宿主机上要建立交叉编译调试的开发环境。环境的建立需要许多的软件模块协同工作，这将是一个比较繁杂的工作，为了便于客户快速入手，华恒科技提供的光盘中带有完整的开发工具及相关说明文档。

嵌入式开发通常要求宿主机配置有网络，支持 NFS（为交叉开发时 mount所用），支持 TFTP（为下载文件时所用）等服务，ubuntu和 Rehhat9中的 NFS和 TFTP服务的配置请见附录部分，其

它版本的 linux请自行到网上查找 tftp及 nfs服务的配置方法。

3.1.2 文件与目录结构

A.光盘目录介绍

华恒提供的光盘一般会包含如下目录：

(1)hharm6410.tgz：整个软件源代码和编译器的压缩包。

(2) arminst.sh：安装脚本文件，可以在 shell提示符下使用 vim进行编辑，进入光盘目录以后，键入 ./arminst.sh就会提示您安装上面提到的 tgz 扩展名的文件，并且安装相应的编译器和拷贝minicom的配置文件到相关目录中去。

(3)Doc：HHARM6410 techmanual-vXX 开发套件的操作及说明手册及芯片手册。

(4) circuit……：开发套件的底板的 PCB图和原理图，其它的电路图。

其它工具目录：

B．安装开发环境软件包：

请您启动 PC上的 Linux操作系统，进入安装光盘，用 root用户权限运行 arminst这个 shell脚本，即可执行安装程序。安装的过程中会显示一些提示信息。通常情况下，您只需按下 y键后，按回车键即可完成整个开发环境的安装。

安装脚本执行以下操作：

1 在根目录下面建立 /HHARM6410目录，并将安装包解压到该目录

2 在 /usr/local/arm目录下面安装编译工具。

3 在 /opt目录下安装编译工具

4 设置串口链接

5 设置 minicom（会把原配置文件备份）

C．安装光盘后的目录介绍

安装华恒提供的光盘以后，会在您的 PC 的根目录下生成“ HHARM6410”等字样的目录，并且把编译器安装到指定的路径中去。以 /HHARM6410为例，HHARM6410目录下的子目录如下：

/HHARM6410/s3c-u-boot-1.1.6/ ：U-Boot 源代码目录，版本是 1.1.6，在该目录下执行 HHTECHmk.uboot即可生成 HHARM6410的 bootloader——u-boot.bin文件；

/HHARM6410/s3c-linux-2.6.21/：Linux 内核源代码目录及驱动源代码；

/HHARM6410/applications/ ：应用程序目录，用户可参考其中的应用程序添加自己的应用；

/HHARM6410/images/ ：目录中是编译好的映像文件或者可执行文件，其中：zImage是编译好的 Linux内核映像文件（和文件系统压缩在一起），u-boot.bin是编译好的引导程序二进制代码，qtopia-4.2.4是编译好的 qt程序 .

/HHARM6410/romfs ：文件系统内容

/HHARM6410/toolchains/ ：编译器的压缩包。

/HHARM6410/minirc.dfl ：华恒开发板启动时默认的串口终端配置 使用第一个串口(ttyS0)。如果您的串口设置出了问题，可以把此文件拷贝到 PC机的 /etc目录，然后退出 minicom，再重新进入 minicom即可。

/HHARM640/multimedia/ ：目录中是编解码和 TVOUT相关的驱动和应用程序：

/HHARM640/multimedia/MFC/ ：包括硬件编解码的驱动和应用程序。

/HHARM640/multimedia/post_processor/：包括图像颜色空间转换和缩放的预处理驱动及应用程序。

/HHARM 640/multimedia/TV_S3C6410/ ：包括 TV输出的驱动及应用程序。

3.1.3 内核编译

安装华恒提供的光盘以后，嵌入式 Linux 内核及设备驱动源代码被安装到 /HHARM 6410/s3c-linux-2.6.21 目录下，交叉编译的工具被放置到 /opt/crosstool/gcc-4.0.1-glibc-2.3.5/ 和 /usr/local/arm 目录下。（下表是对于 HHS3C6410-R1，对于 HHS3C6410-R2 其编译器在 /opt/arm-2008q3/bin/下）

GNU工具集		
arm-unknown-linux-gnu-gcc	arm-unknown-linux-gnu-cc1plus	aarm-unknown-linux-gnu-ranlib
arm-unknown-linux-gnu-as	arm-unknown-linux-gnu-objcopy	arm-unknown-linux-gnu-strings
arm-unknown-linux-gnu-ld	arm-unknown-linux-gnu-objdump	arm-unknown-linux-gnu-gdb
arm-unknown-linux-gnu-g++	arm-unknown-linux-gnu-strip	arm-unknown-linux-gnu-gasp
arm-unknown-linux-gnu-cc1	arm-unknown-linux-gnu-nm	arm-unknown-linux-gnu-size
arm-unknown-linux-gnu-cpp	arm-unknown-linux-gnu-ar	arm-unknown-linux-gnu-addr2line

直接在 /HHARM 6410/s3c-linux-2.6.21目录下执行：

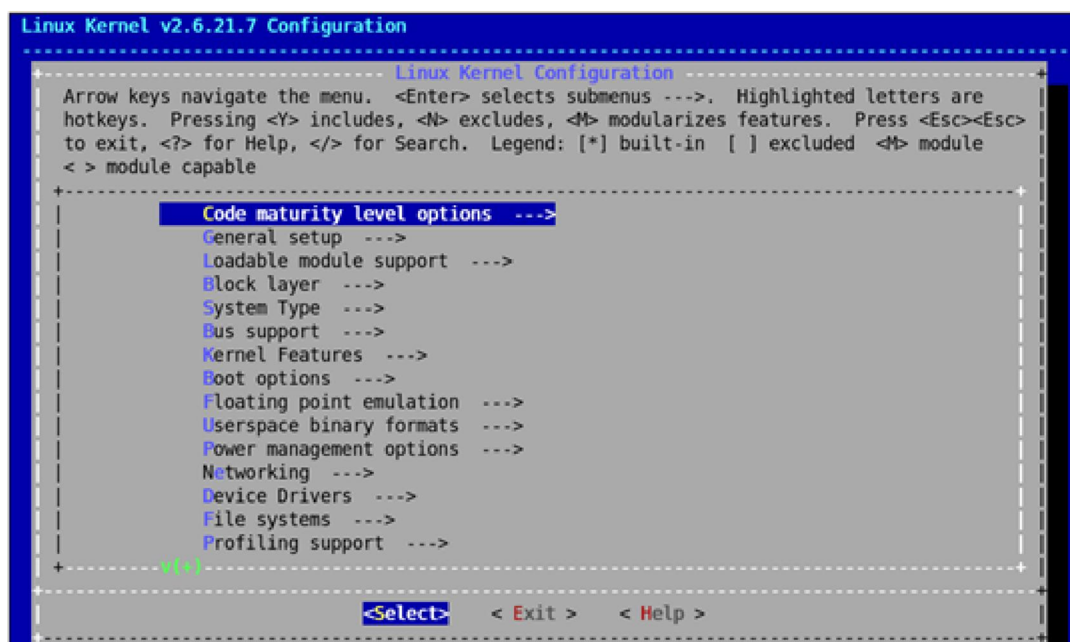
make

即可编译内核，编译完内核以后，会自动把新生成的 zImage 复制到 /tftpboot目录下（以供后面的 tftp命令下载）。

如果需要对内核进行配置和裁剪，可以在 /HHARM 6410/s3c-linux-2.6.21目录下键入

make menuconfig

则出现如下界面，可逐项对内核和驱动模块进行选择和配置（可见内核版本为：Linux Kernel v2.6.21.7）：



完成配置后，退出，保存配置，执行 make 即可编译生成自己定制的内核映像文件 zImage。

华恒科技提供配置内核的默认配置文件 hharm6410.config，如果把内核配置乱了，可以在 /HHARM 6410/s3c-linux-2.6.21 目录下使用如下命令把我们提供的默认配置文件替换出错的内核配置文件 .config。

cp hharm6410.config .config
如果想了解编译的整个过程，可键入如下命令把编译过程重定向到 log文件，进行分析观察：
make &>log //把编译信息输出到文件 log中
vim log

3.1.4 烧制内核映像和文件系统

客户拿到手的开发板，华恒公司已经把相关代码烧写进去了，开发板已经可以启动了，如果需要对内核和文件系统做出修改，则需要重新烧写 Linux 内核与文件系统映像文件 zImage(注意，HHARM 6410平台，我们把内核和文件系统压缩成一个文件了)。

◆※注释※◆

minicom: linux下的一个终端程序，就像 windows中的超级终端一样，只需要在 Shell提示"#"下键入 minicom就可以进入 minicom终端了，更改相应的设置可以按 Ctrl+a, o进入配置，退出 minicom请按 Ctrl+a,q，详细说明，参考 minicom的帮助文件和手册。

◆※注释※◆

下载及烧写命令列表：

setenv serverip xxx.xxx.xxx.xxx //设置 tftp服务器 IP地址
tftp 50008000 zImage //通过 TFTP下载内核
nand erase 40000 700000 //擦除 flash,擦除长度(大小)比 zImage文件大小稍大。
nand write 50008000 40000 700000 //烧写刚下载的文件到指定的位置。

烧写的详细过程如下：

A．按一下复位键，在 minicom 中按一下空格键让开发板停留在"SMDK6410#"提示符下
串口输出如下：(只作为参考)

```
U-Boot 1.1.6 (Jul  7 2008 - 10:52:29) for SMDK6410
CPU:     S3C6410@533MHz
         Fclk = 533MHz, Hclk = 133MHz, Pclk = 66MHz (ASYNC Mode)
Board:   SMDK6410
DRAM:    128 MB
Flash:   0 kB
NAND:    64 MB
In:      serial
Out:     serial
Err:     serial
Hit any key to stop autoboot:  0
```

SMDK6410 # //需要立即按空格键，让其停留在此提示符下。否则其会一直向下引导，若这样，可以按一下复位键，再次立即按一下空格键进入此提示符。

B．设置环境变量。

SMDK6410 #setenv serverip 192.168.2.111

//设置 TFTP 服务器的地址，如果不设置，开发板就会使用 U-Boot中默认的 TFTP服务器的 IP，当然，如果您的 PC机的 IP正好是 U-Boot要求的 IP地址，那就不用执行此命令。

SMDK6410 #**setenv ipaddr 192.168.2.222**

//设置开发板的 IP地址，如果不设置，开发板就会使用 U-Boot中默认的 IP。

SMDK6410 #**setenv gatewayip 192.168.2.1**

//设置网关 ip 地址，如果不设置，开发板就会使用 U-Boot中默认的网关的 IP。

C.保存环境变量

SMDK6410 #**saveenv**

//保存环境变量，这样，即使 PC 机的 IP 与 U-Boot 要求的不一致，通过上面的设置，下次再下载文件时，所要求的 tftp服务器的 IP地址就是上面使用 setenv命令设置的 IP了。

D. 通过 TFTP 下载 Linux 内核映像文件 zImage (其实此文件包含了 linux内核及文件系统两个部分)

SMDK6410 #**tftp 50008000 zImage**

串口输出如下：(只作为参考)

```
dm9000 i/o: 0x18000300, id: 0x90000a46
MAC: 00:22:12:34:56:90
operating at 100M full duplex mode
TFTP from server 192.168.2.111; our IP address is 192.168.2.222
Filename 'zImage'.
Load address: 0x50008000
Loading:
#################################################################################################################################################################
done
Bytes transferred = 6870984 (68d7c8 hex)
```

【注意】!括号中的数值表示刚下载文件的大小(十六进制值)，下面使用 nand erase和 nand write命令烧写时最后一个参数的值要比这个值大。

E. 擦除 flash、烧写刚下载的内核映像文件

SMDK6410 #**nand erase 40000 700000**

串口输出如下：(只作为参考)

```
NAND erase: device 0 offset 0x40000, size 0x700000
Erasing at 0x73c000 -- 100% complete.
OK
```

SMDK6410 #**nand write 50008000 40000 700000**

```
NAND write: device 0 offset 0x40000, size 0x700000
 7340032 bytes written: OK
```

【注意】!很多用户在烧写时都不注意 nand write命令后面跟的三个参数的意义。其中十六进制值 700000比刚才使用 tftp下载时最后一行提示信息的括号中的 zImage文件大小 68d7c8 hex要大，

否则，只烧写了刚下载文件的部分，后面引导就会出现问题。

『说明』

A. 以上下载都是通过 tftp服务来完成的，宿主机必须把 TFTP服务器配置好，一般情况下 tftp命令默认从 TFTP服务器的 /tftpboot目录下载文件到开发板的 RAM 中的，要保证所指定的 TFTP服务器的 /tftpboot目录下存在要下载的文件，否则下载时会出现"File not Found"这样的提示，可以把 HHARM 6410 /images目录下华恒科技出厂时可以使用的文件拷贝到 TFTP服务器的 /tftpboot目录下以供 tftp下载。

B. nand erase命令后的两个参数的意义为：第一个参数是要檫除的 nand flash的起始地址；第二个参数是要檫除的大小。

C. nandwrite命令后的三个参数的意义为：第一个参数是刚才通过 tftp把文件下载到开发板中 RAM 中的地址；第二个参数是烧写到 Flash的地址；最后一个参数表示在烧写指定的文件时，在 flash中给这个文件的空间大小，此参数的值一定要比刚才通过 tftp命令下载的文件的大小要大，否则，刚才的文件只烧录了一部分，没有全部烧录进去，就会造成问题。此值是根据 tftp命令下载时提示的文件的大小来决定的，在通过 tftp命令下载的时候，最后会提示您（十六进制值的）文件的大小，通常情况下，保持最后 4个数字为"0"就可以了。可以在 U-Boot提示符 (SMDK6410 #)下键入 help查看 nandwrite命令的帮助。

D. 首先通过以太网从指定的 TFTP 服务器上下载指定的文件到内存中，然后把内存中的内容烧写到 flash。

E. tftp命令和 nandwrite命令后的所有十六进制数值在键入命令时，0x前缀可以省略。

F. 如果把开发板和 PC机的以太网口直接相连，要有交叉对接网线 (1,3；2,6交换），如果要用普通的网线，则要经过网络连接设备 (集线器，交换机等等），建议使用对接网线。

G. U-Boot的提示符 (SMDK6410 #)下，若已经键入过命令，按回车表示确认键入的命令，若再次按下回车，则重复执行前一条命令。

H. 在 U-Boot提示符 (SMDK6410 #)下键入 help查看命令的帮助。

I. 在 /HHARM 6410/images/HHTECH-burn-cmd文件中，保存了下载和烧写时的命令，以方便烧写时使用。

J. 在 U-Boot提示符下，多个命令可以写到一行上，各个命令之间使用分号间隔。

K. 在 U-Boot提示符下，使用 setenv bootcmd命令时注意要加引号，例如：

setenv bootcmd "setenv serverip 192.168.2.187;tftp 50008000 zImage;bootm"就设置开发板上电后从 192.168.2.187的宿主机处下载 zImage并启动。

『说明』

用户可以修改 U-Boot中指定的 TFTP服务器和开发板的 IP地址，方法有两种。

1. 不用重新烧写 u-boot.bin，在"SMDK6410 #"提示符下，键入以下命令，即可把您刚设置的 IP保存在 flash中。

SMDK6410 # **setenv serverip 192.168.3.222** //设置 TFTP服务器的 IP

SMDK6410 # **setenv ipaddr 192.168.3.111** //设置开发板网口的 IP

SMDK6410 # **saveenv**

重新启动开发板，以后下载文件时，指定的 TFTP服务器的 IP就改成您刚设置的 IP了，这样并没有重新烧写 u-boot.bin文件。

重新启动开发板，则下次通过 tftp下载文件时，所要求的 tftp服务器的 IP就为 192.168.3.222了。在设置之前和设置之后，都可以通过在 U-Boot的提示符下键入 printenv命令来查看设置的环境变量。

SMDK6410#**printenv**

串口输出如下：（只作为参考）

```
bootcmd=nand read 50008000 40000 700000;bootm 50008000
bootdelay=3
baudrate=115200
netmask=255.255.255.0
serverip=192.168.2.155
ipaddr=192.168.2.222
gatewayip=192.168.2.1
ethaddr=00:55:5c:26:0a:5b
stdin=serial
stdout=serial
stderr=serial

Enviroment size: 238/16380 bytes
```

2. 修改设置 TFTP服务器的 IP源文件，重新烧写 u-boot.bin文件。

vi /HHARM6410/s3c-u-boot-1.1.6/include/configs/smdk6410.h

修改如下一行：

#define CONFIG_SERVERIP 192.168.2.111

然后运行 s3c-u-boot-1.1.6 目录下的 HHTECH mk.uboot，重新编译 U-Boot，重新烧写新的 u-boot.bin即可。

3.1.5 在 RAM 中运行内核

为了不至于频繁的烧写内核和文件系统，可以把内核和文件系统先下载至 RAM 中直接启动（而不是从 flash读到 RAM 中运行），测试一下新的内核和文件系统是否可以正常使用，如果确定最终需要烧写的文件，再进行烧写也不迟。这一功能可以极大的方便内核调试，其使用步骤如下：

1. 按复位键重启开发板，在 minicom中应该有启动信息，立即按空格键，让开发板停留在 U-Boot 的提示符 " SMDK6410 #"；

2. 下载运行内核 zImage：

SDMK6410# **tftp 50008000 zImage ; bootm**

即可直接启动刚下载的内核，并最终进入 shell提示符 " #"。

这样做的好处是不用频繁的烧写 flash，可以延长 flash的使用寿命，推荐用户在调试自己编译的内核、加入自己的应用程序的时候使用此方法，等确定最终烧写的文件时，再烧写到 flash中去。

3.1.6 启动烧制的嵌入式的 LINUX 系统

烧写完 U-Boot，zImage以后，上电复位或硬件 RESET的方式重新启动目标板，或者在 U-Boot 提示符下输入命令启动 Linux。

SMDK6410# **reset**

通过串口终端可以看到开发板上 U-Boot自检、引导内嵌 Linux 操作系统打印到串口的信息和 Linux的整个启动信息。

启动信息如下：

```
root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-linux-2.6.21
root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-u-boot-1.1.6    root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-linux-2.6.21
U-Boot 1.1.6 (Jul  7 2008 - 10:52:29) for SMDK6400


CPU:     S3C6400@533MHz
         Fclk = 533MHz, Hclk = 133MHz, Pclk = 66MHz (ASYNC Mode)
Board:   SMDK6400
DRAM:    128 MB
Flash:   0 kB
NAND:    64 MB
In:      serial
Out:     serial
Err:     serial
Hit any key to stop autoboot:  0

NAND read: device 0 offset 0x40000, size 0x700000
 7340032 bytes read: OK
Boot with zImage

Starting kernel ...

Uncompressing Linux..............................................................................
Linux version 2.6.21.5-cfs-v19 (root@cc-desktop) (gcc version 4.0.1) #91 Fri Aug 22 10:13:56 CST 208
CPU: ARMv6-compatible processor [410fb761] revision 1 (ARMv6TEJ), cr=00c5387f
Machine: SMDK6400
Ignoring unrecognised tag 0x00000000
Memory policy: ECC disabled, Data cache writeback
CPU S3C6400 (id 0x36400100)
S3C6400: core 533.000 MHz, memory 133.250 MHz, peripheral 66.625 MHz
S3C6400: EPLL 192.000 MHz
S3C24XX Clocks, (c) 2004 Simtec Electronics
```

```
root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-linux-2.6.21
root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-u-boot-1.1.6    root@cc-desktop: /home/cc/6400-ok/HHARM6400/s3c-linux-2.6.21
eth0: link up, 100Mbps, full-duplex, lpa 0xCDE1
WM8987: 0x3343, R25 <= 0x143

Please press Enter to activate this console. mmc0: host does not support reading read-only switch. .
mmc: major= 254, minor= 0
mmcblk0: mmc0:b368 SD     1997312KiB
 mmcblk0: p1

starting pid 260, tty '': '-/bin/sh'
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:22:12:34:56:90
          inet addr:192.168.2.222  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:20 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:3965 (3.8 KiB)  TX bytes:0 (0.0 B)
          Interrupt:74 Base address:0x300

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

# ping 192.168.2.156
PING 192.168.2.156 (192.168.2.156): 56 data bytes
64 bytes from 192.168.2.156: seq=0 ttl=128 time=4.350 ms
```

3.1.7 利用 U-Boot 更新 U-Boot 自身

『说明』：用户在使用开发板的过程中，若遇到系统崩溃、修改了引导代码，就要考虑重新烧写 U-Boot 在没有上述情况时，无须重新烧写 U-Boot 开发板在出厂时华恒科技已经烧写了 U-Boot 引导代码无法启动，下载、烧写内核及文件系统等工作都无法进行。

『说明』在嵌入式系统中，通常把引导系统的初始化部分的代码统称为 bootloader，相当于 PC 机的 BIOS。在华恒科技提供的各种嵌入式开发平台中（ARM 系列套件、PowrPC系列套件，Coldfire 系列套件、ADSP系列套件等，MIPS系列套件），引导代码有的使用 U-Boot 有的是 PPCBoot 有的是 Redboot等等，但实际烧写到 flash中的文件一般为 u-boot.bin ppcboot.bin 的 Redboot.bin等二进制代码文件。

当需要对 U-Boot本身进行修改的时候，一种方式是通过仿真器进行烧写，另一种方式是在 U-Boot引导以后，通过 tftp下载最新的 u-boot.bin后，使用 U-Boot提供的命令进行烧写，像烧写内核和文件系统一样，只是烧写的位置不同而已，实现对自身更新。

通过网络下载 u-boot.bin

```
SMDK6410 #tftp 50008000 u-boot.bin
```

串口输出（只作为参考）

```
dm9000 i/o: 0x18000300, id: 0x90000a46
MAC: 00:22:12:34:56:90
operating at 100M full duplex mode
TFTP from server 192.168.2.111; our IP address is 192.168.2.222
Filename 'u-boot.bin'.
Load address: 0x50008000
Loading: #################################################
done
Bytes transferred = 196608 (30000 hex)
SMDK6410#nand erase 0 40000
SMDK6410 #nand write 50008000 0 40000
```

利用仿真器烧写 u-boot.bin的过程请见本手册的附录部分。

3.2 软件应用开发

3.2.1 开发模式

宿主机是一台运行 LINUX 的 PC 机，目标板即华恒 HHARM 6410开发板。应用程序的开发有两种模式：

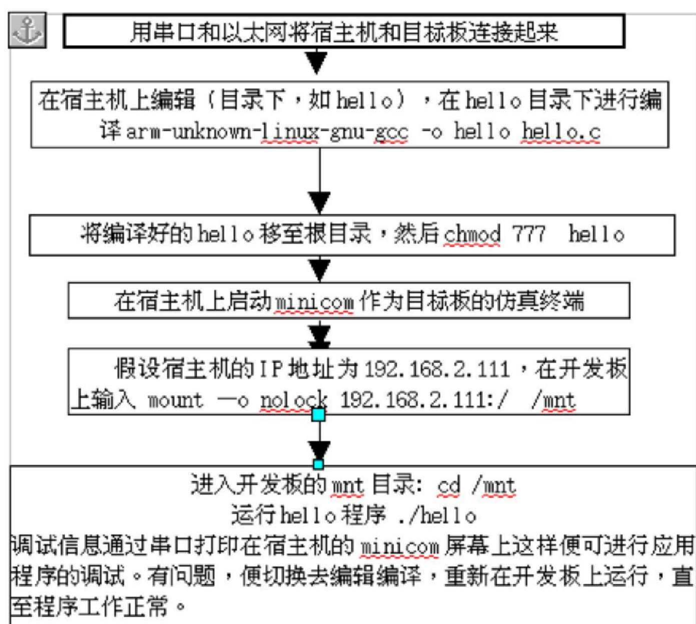
1.先在宿主机（ Intel CPU）上调试通过后，再移植到目标板（HHARM 6410）上。

移植的工作包括两个方面：

A. 函数库的问题。在程序移植时可能会有函数未定义的问题。对于这种问题，一般要求开发者自己编制这些要用到却又未定义的函数。

B. 修改 Makefile以选择适合目标板的编译工具

2.直接在目标板上进行开发（通用开发模式，建议采用该模式）。将宿主机和目标板通过以太网连接，在宿主 PC 机上运行 minicom 作为目标板的显示终端，在目标板上通过 NFS（网络文件系统）把宿主机共享出来的目录 mount上以后，此共享目录也成了开发板的目录了，直接在开发板上运行共享目录下的程序，如果出错，再到 PC机终端下去修改，再编译、调试。下面给出这种直接 TARGET模式下的开发流程：



『说明』：

1.以上说明中，把宿主机的根目录(/)通过NFS共享出去了，但您的机器在配置NFS时，不一定共享了根目录，假如您共享了宿主机下的/Share，则可以使用如下命令挂载此目录到开发板上。

mount –o nolock 192.168.2.111:/Share /tmp 把宿主机上NFS共享出来的/Share目录挂载到开发板的/tmp目录下。

2.要正确理解NFS，宿主机通过NFS共享出来的目录，一旦通过mount挂载到开发板上，则开发板上就多了这样一个资源(本目录及其子目录下的所有文件)，所以，在PC机上所修改的、甚至交叉编译生成的新的文件，则开发板上也可以立即得到。说到底，其实就是PC机把某个资源共享出去了，只要宿主机给您什么权限(读取，删除，执行)，开发板的终端下就有操作某共享资源的权限。当然，以上是PC机把资源共享出去，而不是开发板把某个资源共享出去。

3.PC机上NFS的配置及TFTP的配置请见附录部分。

3.2.2 如何创建编译自己的应用

嵌入式Linux软件开发和普通的Linux软件开发过程类似，需要注意的是交叉编译，主要是选择正确的交叉编译工具和正确的链接库。

Linux下的应用程序大多数都是用C语言开发的。用C语言开发应用程序，首先遇到的问题就是C库的问题，对于HHARM6410这种有MMU的处理器平台上的嵌入式Linux而言，和PC机上使用的Linux一样，用的都是glibc，只不过是使用不同的编译器编译程序而已。

HHARM6410开发包提供了编译好的glibc库的动态库(.so文件)和静态库(.a文件)，默认安装在/opt/crosstool/gcc-4.0.1-glibc-2.3.5/arm-unknown-linux-gnu/arm-unknown-linux-gnu/lib目录下，在应用程序的Makefile文件中指定了编译器的路径后，就会自动链接这些库文件。Makefile文件的格式和内容可以参考/HHARM6410-R1/applications/下某个目录中的Makefile。

下面举例说明：

创建hello.c:

#**cd /HHARM6410/applications/hello**

#**vi hello.c**

输入如下内容：

```
#include <stdio.h>

int main()
{
      printf("Hello World!\n");
      return 0;
}
```

该程序只是向串口打印 Hello world!字符串。编写 Makefile 文件要用 arm-none-linux-gnueabi-gcc这个交叉编译器，用此编译器编译出来的文件，才能在 HHARM 6410 平台上运行。以下是它的 Makefile文件 (用户需要了解 Makefile的一些基础知识，附录部分有介绍)：

```
CC = /opt/arm-2008q3/bin/arm-none-linux-gnueabi-gcc
CFLAGS = -DDEBUG -D__Linux__ -g

all:hello
hello:hello.o
        $(CC) $(CFLAGS) -o $@ $<
clean:
        rm -rf *.o hello
```

然后在 hello目录下执行 make，便生成了可执行文件 hello，可将生成的可执行文件加入 Linux 文件系统中：

#**cp hello /HHARM6410/romfs/hhtech/app**

重新编译内核，烧写新的内核 zImage到开发板上以后，就可以到 hhtech/app目录下运行 hello 文件了。

『说明』：

制作内核与文件系统映像的过程见下一节，另外上面最后一步的过程可以通过 NFS或其他方法 (usb，sd卡等)将应用程序在开发板上执行。

3.2.3 制作文件系统映像

所谓的文件系统就是用户程序和文件的集合。制作文件系统一般是在 Linux PC 机上把用户程序和文件放到一个统一的目录下，然后通过一个工具把此目录打包成一个二进制映象 (image)文件，这通常是个压缩的过程。

Linux 内核启动完成后，会挂接文件系统映象文件并做解包的工作，当然第一个挂接的就是根文件系统 (rootfs)了，有了 rootfs并提供了 SHELL之后，就可以通过 SHELL命令“mount”来挂载其它文件系统 (FS)了。所以，若用户要添加自己的程序到文件系统的 image里面去，就是把自己的程序或文件复制到这个文件系统在 PC 机上的相应目录里面，然后重新打包生成 image，这样你的程序 (或文件)就加进去了，以后就是下载和烧写的问题了。

那么具体在做“把自己的程序或文件复制到这个 FS在 PC Linux上的相应目录里面去”这个步骤的时候，不同的文件系统就略有区别了。

HHARM 6410使用 initramfs文件系统，编译内核的时候会自动把内核与文件系统压缩在一起。

/HHARM 6410/romfs为 Linux的文件系统映像文件目录。

在 /HHARM6410/s3c-linux-2.6.21目录下 make menuconfig配置：
General setup --->

```
[*] System V IPC
[ ]   IPC Namespaces
[ ] POSIX Message Queues
[ ] BSD Process Accounting
[ ] Export task/process statistics through netlink (EXPERIMENTAL)
[ ] UTS Namespaces
[ ] Auditing support
< > Kernel .config support
[*] Create deprecated sysfs files
[ ] Kernel->user space relay support (formerly relayfs)
[*] Initial RAM filesystem and RAM disk (initramfs/initrd) support
(./romfs) Initramfs source file(s)
(0)     User ID to map to 0 (user root)
(0)     Group ID to map to 0 (group root)
[*] Optimize for size (Look out for broken compilers!)
[ ] Configure standard kernel features (for small systems)  --->
```

其中高亮的一行就是选择文件系统的路径。加入用户程序，只需

cd /HHARM6410/romfs

cp hello hhtech/app

然后就是使用如下命令，生成新的内核文件 zImage

cd /HHARM6410/s3c-linux-2.6.21

make

下载烧写新的 zImage完毕后，重启开发板。

#**ls hhtech/app**

asp_test cam2fb key mp3play netperf ossrec record ttytest usbcamerer_test
bonnie++ hello mfc-display mplayer ossplay post_test runqpe tv_test

目录结构中出现了 hello应用程序

运行文件：

./hello

HelloWorld! //运行结果

也许您想让 hello文件在开发板启动以后自动执行，不需要每次手动键入命令运行，像华恒科技提供的开发板的 IP设置、LCD上显示的 QT以及一些自动加载驱动一样，那就把 /hhtech/app/hello这一行命令添加到开发板根文件系统的启动脚本 /etc/init.d/rcS里

busybox简介

BusyBox 最初是由 Bruce Perens 在 1996 年为 Debian GNU/Linux 安装盘编写的。其目标是在一张软盘上创建一个可引导的 GNU/Linux 系统，这可以用作安装盘和急救盘。一张软盘可以保存大约 1.4-1.7MB 的内容，因此这里没有多少空间留给 Linux 内核以及相关的用户应用程序使用。

BusyBox 是很多标准 Linux® 工具的一个单个可执行实现。BusyBox 包含了一些简单的工具，例如 cat和 echo，还包含了一些更大、更复杂的工具，例如 grep、find、mount以及 telnet(不过它的选项比传统的版本要少)；有些人将 BusyBox 称为 Linux 工具里的“瑞士军刀”。

BusyBox 揭露了这样一个事实：很多标准 Linux 工具都可以共享很多共同的元素。例如，很多基于文件的工具（比如 grep 和 find）都需要在目录中搜索文件的代码。当这些工具被合并到一个可执行程序中时，它们就可以共享这些相同的元素，这样可以产生更小的可执行程序。实际上，

BusyBox 可以将大约 3.5MB 的工具包装成大约 200KB 大小。这就为可引导的磁盘和使用 Linux 的嵌入式设备提供了更多功能，节省宝贵的内存空间。

在实际使用中，通常把 busybox放在根文件系统的 bin目录下，bin和 sbin目录下的很多可执行文件其实都是链接到 busybox，busybox到底支持哪些命令，可以在 busybox的源代码目录下进行配置，新的 busybox支持 make menuconfig进行配置，配置完以后，把新生成的 busybox替换掉根文件系统中原来的 busybox，相对于 PC机上的相关命令，busybox上支持的同样的命令一般情况下都作了简化。

Busybox源代码华恒科技一般放在 /HHARM6410/applications/busybox目录下：

#**cd /HHARM6410/applications/busybox**

#**make menuconfig**

进入 Busybox Settings --->

Installation Options --->

可以看到：

```
[*] Don't use /usr
    Applets links (as soft-links)  --->
(../../romfs) BusyBox installation prefix
```

所以执行 make;make install后会自动更新文件系统中的 busybox，然后再按照前面的方法更新内核与文件系统映像即可。

3.2.4 Qt 的移植

Qt移植参见附录 E。

第四章 外围接口使用介绍

为了测试方便，华恒科技在文件系统的根目录下建立了 hhtech这样的目录，并在其下建立相关子目录，执行 /hhtech/app目录下的相关程序，就可以对外围接口进行测试。

4.1 USB HOST

驱动程序文件位置： /HHARM6410/s3c-linux-2.6.21/drivers/usb/host

HHARM6410底板提供了一个 USB Host接口，可支持 U盘，USB摄像头，USB无线网卡，蓝牙适配器等多种 USB 设备，但使用不同的设备需要开发相应的设备驱动。华恒标准套件提供了 U盘的驱动程序，用户可直接使用 U盘。

内核启动后可以看到信息：

```
s3c2410-ohci s3c2410-ohci: S3C24XX OHCI
s3c2410-ohci s3c2410-ohci: new USB bus registered, assigned bus number 1
s3c2410-ohci s3c2410-ohci: irq 47, io mem 0x74300000
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 2 ports detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
```

启动完后，插入 U盘时，会出现如下提示：

```
~ #
# usb 1-1: new full speed USB device using s3c2410-ohci and address 2
usb 1-1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
scsi 0:0:0:0: Direct-Access     USB 2.0  SD/MMC Reader         PQ: 0 ANSI: 0 CCS
SCSI device sda: 16007168 512-byte hdwr sectors (8196 MB)
sda: Write Protect is off
sda: assuming drive cache: write through
SCSI device sda: 16007168 512-byte hdwr sectors (8196 MB)
sda: Write Protect is off
sda: assuming drive cache: write through
 sda: sda1
sd 0:0:0:0: Attached scsi removable disk sda
sd 0:0:0:0: Attached scsi generic sg0 type 0
```

执行如下命令查看分区信息：

```
#cat /proc/partitions
major minor  #blocks  name
   8     0    8003584 sda
   8     1    7999488 sda1
```

挂载 u盘：

~ #mount -t vfat /dev/sda1 /tmp/

~ #cd /tmp/

/tmp # ls

1mp3	ffplay	rc-mp3	rc-recorder2
1.rar	mp3play	rc-mtv	rc-video
2.rar	mplayer	rc-recorder	v120x160.avi

支持多种文件系统，如：EXT2，VFAT等。

内核编译选项为：

#cd /HHARM6410/s3c-linux-2.6.21

#make menuconfig

进入 Device Drivers --->

SCSI device support --->

选择如下选项：

```
<*> SCSI device support
< >   SCSI target support
[*] legacy /proc/scsi/ support
--- SCSI support type (disk, tape, CD-ROM)
<*> SCSI disk support
< > SCSI tape support
< > SCSI OnStream SC-x0 tape support
< > SCSI CDROM support
<*> SCSI generic support
```

进入 Device Drivers --->

USB support --->

```
<*> Support for Host-side USB
[ ]   USB verbose debug messages
---   Miscellaneous USB options
[*]   USB device filesystem
[ ]   Dynamic USB minor allocation (EXPERIMENTAL)
---   USB Host Controller Drivers
< >   ISP116X HCD support
<*>   OHCI HCD support
< >   SL811HS HCD support
< >   USB OTG Host support
--- USB Device Class drivers
< > USB Modem (CDC ACM) support
< > USB Printer support
--- NOTE: USB_STORAGE enables SCSI, and 'SCSI disk support'
--- may also be needed; see USB_STORAGE Help for more information
<*> USB Mass Storage support
```

4.2 USB Device

USB Device 驱动位于 /HHARM6410/s3c-linux-2.6.21/drivers/usb/gadget 目录下，主文件为 s3c-udc-otg-hs.c

HHARM6410的核心板有一个 usb device接口，并提供 usb device的存储设备功能。

内核配置选项如下：

#cd /HHARM6410/s3c-linux-2.6.21

#make menuconfig

进入 Device Drivers --->

USB support --->

USB Gadget Support --->

选择如下选项：

```
<*> Support for USB Gadgets
[ ]   Debugging information files
    USB Peripheral Controller (S3C high speed(2.0, dual-speed) USB OTG device)
      S3C high speed(2.0, dual-speed) USB OTG device
<M> USB Gadget Drivers
< >   Gadget Zero (DEVELOPMENT)
<M>   Ethernet Gadget (with CDC Ethernet support)
[*]     RNDIS support (EXPERIMENTAL)
< >   Gadget Filesystem (EXPERIMENTAL)
<M>   File-backed Storage Gadget
[*]     File-backed Storage Gadget testing version
< >   Serial Gadget (with CDC ACM support)
< >   MIDI Gadget (EXPERIMENTAL)
```

然后编译模块：

make modules

make modules_install

生成新的映像文件：

make

烧写新的 zImage并启动开发板后可以看到编译好的模块：

#ls /lib/modules/2.6.21.5-cfs-v19/kernel/drivers/usb/gadget/

g_ether.ko g_file_storage.ko

加载 g_file_storage.ko模块需要通过 file参数指定需加载的文件，可以创建一个文件：

#dd if=/dev/zero of=10m bs=1M count=10

然后加载驱动：

#modprobe g_file_storage file=10m stall=1 removable=1

可以看到驱动成功加载，然后用 usb线连接核心板上的 usb device口和 PC机的 usb口（最好在 windows下测试，Linux下的支持不是很好），可以看到 windows提示找到磁盘，把盘格为 fat格式，就可以读写这个磁盘了。可以往磁盘中拷贝一些文件，卸载磁盘后在开发板上执行：

#mount -o loop 10m /mnt

#ls /mnt

查看拷贝的文件是否存在。

file参数也可以指定块设备节点，比如开发板插上 sd卡并找到后会创建 /dev/mmcblk0p1节点，这时可以加载 g_file_storage驱动：

#modprobe g_file_storage file=/dev/mmcblk0p1 stall=1 removable=1

windows下就找到新磁盘，可以测试读写。

4.3 LCD/触摸屏

LCD驱动程序主文件：

/HHARM6410/s3c-linux-2.6.21/drivers/video/s3c_hhtech.c

/HHARM6410/s3c-linux-2.6.21/drivers/video/s3cfb.c

触摸屏驱动程序主文件：

/HHARM6410/s3c-linux-2.6.21/drivers/input/touchscreen/s3c-ts.c

华恒 HHARM6410标准套件中配置了 240×320 TFT彩屏 LCD（带触摸屏），提供 framebuffer驱动，可对 QT等嵌入式图形系统提供良好的支持。

此外还可定制支持各种其它的屏，例如 10.4寸 640×480的大屏，3.5寸 320x240等。

套件还提供了一写测试例程，如：ts_test，ts_print 例程源代码在/HHARM6410/applications/tslib-1.0目录下。

LCD的支持需要对内核进行配置，内核编译选项：

make menuconfig

进入 DeviceDrivers --->

Graphics support --->

选择如下选项：

```
<*> Support for frame buffer devices
[*]    Enable firmware EDID
[*]    Enable Video Mode Handling Helpers
[ ]    Enable Tile Blitting Support
---    Frame buffer hardware drivers
<*>    S3C SMDK LCD framebuffer support
         Select LCD Type (HHTECH FB)  --->
<*>      Advanced low level driver options
           Select BPP(Bit Per Pixel) type (16 BPP)  --->
           Choose postprocessing  (Postprocessing NOT supported)  --->
(2)        Number of S3C FB windows
           Choose virtual screen support  (Virtual screen NOT supported)  --->
           Choose double buffering support  (double buffering NOT supported)  -
```

测试 LCD及触摸屏：

启动开发板以后，执行以下命令就可以在 LCD上看到触摸屏的示例程序了：

/hhtech/app/ts_test

Qt测试：

/hhtech/app/runqpe

4.4 100M 以太网

驱动程序主文件 ：/HHARM6410/s3c-linux-2.6.21/drivers/net/dm9000.c

测试程序文件位置：/HHARM6410/applications/netperf-2.4.1/

HHARM6410通过外接一片 DM9000以太网 MAC芯片扩展了一个 10/100M自适应的以太网接口，占用资源：nGCS1/EINT10.

内核编译选项为：

make menuconfig

进入 DeviceDrivers --->

Network device support --->

Ethernet (10 or 100Mbit) --->

选择如下选项：

```
[*] Ethernet (10 or 100Mbit)
---   Generic Media Independent Interface device support
[ ] 3COM cards
[ ] Western Digital/SMC cards
< > SMC 91C9x/91C1xxx support
<*> DM9000 support
[ ] Racal-Interlan (Micom) NI cards
< > AT1700/1720 support (EXPERIMENTAL)
< > DEPCA, DE10x, DE200, DE201, DE202, DE422 support
< > HP 10/100VG PCLAN (ISA, EISA, PCI) support
[ ] Other ISA cards
[ ] EISA, VLB, PCI and on board controllers
```

测试步骤：

先在 PC Linux 上编译 netperf，得到可在 PC上运行的 netserver和 netperf，然后交叉编译 netperf 得到可在开发板上运行的 netserver和 netperf。开发板用网线（对接线）同 PC机相连；

PC机上运行

./netserver

在 minicom下执行客户端程序（开发板上）

/hhtech/app/netperf -H 192.168.2.111

其中 192.168.2.111为 PC机的 IP。

TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.2.111 (192.168.2.111) port 0 AF_INET

```
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec
 87380  16384  16384    10.00      44.20
```

速度大约为 5.5MB/s

同样可在开发板上运行 netserver，PC上运行 netperf测试网速。

4.5 视频输入（可选 ov9650 或 saa7113)

驱动程序：

/HHARM6410/s3c-linux-2.6.21/drivers/media/video/s3c_camera_driver.c
/HHARM6410/s3c-linux-2.6.21/drivers/media/video/s3c_camif.c
/HHARM6410/s3c-linux-2.6.21/drivers/media/video/samsung/ov9650.c
/HHARM6410/s3c-linux-2.6.21/drivers/media/video/samsung/saa7113h.c

内核配置：

make menuconfig

进入 Device Drivers --->

Multimedia devices --->

选择：

```
<*> Video For Linux
[*]    Enable Video For Linux API 1 (DEPRECATED)
---    Enable Video For Linux API 1 compatible Layer
       Video Capture Adapters  --->
       Radio Adapters  --->
    Digital Video Broadcasting Devices  --->
< > DABUSB driver
```

进入 Video Capture Adapters --->

选择：

```
[*] S3C SMDK CAMERA support
       Select CIS module type (OV9650 decoder)  --->
    V4L USB devices  --->
```

在 Select CIS module type

开发板起来后，运行

#/hhtech/app/cam2fb /dev/video12

或者

#/hhtech/app/cam2fb /dev/video13

就能在 LCD上看到摄像头捕捉到的图像。

『说明』

关于选择 SAA7113输入：

需要在 SAA7113芯片旁边 J20跳线帽插上（使用 OV9650请移除），然后在 make menuconfig 选择 7113，接上模拟摄像头即可。测试方法与 OV9650相同。如果只使用模拟视频做为输入，为了获得更好的图像质量建议将跳线 J20换成在其背后焊接 33欧姆电阻。

cam2fb程序说明：

cam2fb需要两个参数，第一个参数指定摄像头设备节点， S3C6410的摄像头接口支持 preview 输出和 codec输出， /dev/video12对应 codec输出， /dev/video13对应 preview输出；第二个参数指定 lcd屏的颜色深度，对应 240x320的 16位屏第二个参数就为 16新版的程序不需要这个参数

cam2fb运行中输入 capt并按回车就会截图并保存为 /tmp/xx.bmp，其中 xx为序号；输入 quit并按回车就会退出应用程序。

4.6 串口通信和 GPRS 拨号（可选）

驱动程序主文件：

/HHARM6410/s3c-linux-2.6.21/drivers/serial/s3c6410.c

测试程序文件：

/HHARM6410/applications/ttytest/

内核配置：

make menuconfig

进入 Device Drivers --->

Character devices --->
Serial drivers --->

选择：

```
< > 8250/16550 and compatible serial support
--- Non-8250 serial port support
< > Samsung S3C2410/S3C2440/S3C2442/S3C2412 Serial port support
<*> Samsung S3C6400 Serial port support
[*]   Support for console on S3C64XX serial port
[ ]   Support for High speed UART(Max.4000000bps) on S3C64XX serial port
```

测试串口通信的步骤（ttytest）：

开发板上的两个串口分别接 PC机的两个串口或接两台 PC机的各一个串口，并分别运行 minicom；在默认的输入输出串口 1所接的 minicom窗口下键入：

#/hhtech/app/ttytest

在串口 2所接 minicom窗口下可看到收到 ttytest程序发过来的字符。在第二个串口接的 minicom中键入字符，在默认的输入输出串口 1的 minicom也可以收到在第二个串口中键入的字符。

此外，串口还有一个用途，就是接 MODEM进行 PPP拨号。

1、首先是根据开发板的串口 2的信号定义制作接 MODEM的串口线，华恒科技提供的开发板的串口就相当于外置的普通的使用电话线拔号上网的 MODEM串口，所以，需要将开发板的“收”信号与 MODEM的“发”信号相连，开发板的“发”信号与 MODEM的“收”信号相连，主要用到的就是四个信号(RX，TX，RTS，CTS)，另外还有地(DB9接头中的 5与 DB9接口的外壳相连即可)。对应到 DB9串口接头上，就是要将接 MODEM的串口线的其中一端 2、3交换，7、8交换。做好线之后，可用 tip软件进行与 MODEM通信的测试：

tip -l /dev/ttyS1 -s 115200

显示 connected后，用户就可以输入 AT命令直接与 MODEM对话了，这样做的目的是测试串口线，如果返回 OK字样，则说明与 MODEM通信基本正常。

2、配置内核支持 PPP，并编译拨号用相关应用程序：pppd/chat。关于内核支持，就是 make menuconfig在内核配置的 Network device support ---> 中选择 PPP/SLIP/CSLIP等，具体细节选项可任选，其实选中以下几项就可以了。

```
<*> PPP (point-to-point protocol) support
    [*]  PPP multilink support (EXPERIMENTAL) (NEW)
    <*>  PPP support for async serial ports (NEW)
    <*>  PPP support for sync tty ports (NEW)
         < >  PPP Deflate compression (NEW)
         < >  PPP BSD-Compress compression (NEW)
       < >  PPP over Ethernet (EXPERIMENTAL) (NEW)
<*> SLIP (serial line) support
[*]  CSLIP compressed headers (NEW)
```

3、编写拨号脚本和相关配置文件。做拨号可以先在安装 Linux的 PC上手工拨号成功（建议不要用图形界面）后，然后再能到开发板上做测试。而且，这只是第一步，简单的拨号上去并没有任何意思，因为产品化过程中，还要涉及许多的应用细节，例如断线重拨检测，开机自动拨号，无流量自动断开，有流量自动拨号等，都是非常麻烦的细节，需要做长时间大量的稳定性可靠性测试。

『说明』

华恒提供经过全球各大城市测试稳定应用的拨号软件，可大大节省用户产品的上市时间。可联

系华恒市场部购买 PPP(Modem/GPRS/CDMA拔号，通过串口 拔号软件包和 PPPoe(ADSL拔号，通过以太网口 拔号软件包，软件包中提供了 PPP和 PPPoe拔号的源代码和说明文档。

4.7 8x2 键盘

驱动位置 :/HHARM6410/s3c-linux-2.6.21/drivers/input/keyboard/s3c-keypad.c
测试应用程序源代码： /HHARM6410/applications/keypad/
测试应用程序在开发板上的位置： hhtech/app/key
key 1 Pressed
key 1 Released
key 2 Pressed
key 2 Released

4.8 音频输出和录音

驱动程序文件位置 ：
/HHARM6410/s3c-linux-2.6.21/sound/soc/s3c/smdk6410_wm8987.c
录音程序文件位置 ：
/HHARM6410/applications/record/
放音程序文件位置：
/HHARM6410/applications/mp3play/
/HHARM6410/applications/MPlayer-0.91/
支持 ADPCM /PCM 录音和 WAV文件播放。

内核配置：

make menuconfig
进入 Device Drivers --->
Sound --->
选择

```
<*> Sound card support
    Advanced Linux Sound Architecture  --->
    Open Sound System  --->
```

进入 Advanced Linux Sound Architecture --->
选择

```
<*> Advanced Linux Sound Architecture
< >   Sequencer support
<*>   OSS Mixer API
<*>   OSS PCM (digital audio) API
[*]     OSS PCM (digital audio) API - Include plugin system
[ ]   Dynamic device file minor numbers
[ ]   Support old ALSA API
[ ]   Verbose procfs contents
[ ]   Verbose printk
[ ]   Debug
      Generic devices  --->
      ALSA ARM devices  --->
      USB devices  --->
      System on Chip audio support  --->
```

进入 System on Chip audio support --->

选择

```
<*> ALSA for SoC audio support
    SoC Audio for the Atmel AT91  --->
    SoC Audio for the Intel PXA2xx  --->
    SoC Audio for the Samsung S3C  --->
```

进入 SoC Audio for the Samsung S3C --->

选择

```
<*> SoC Audio for the Samsung S3C chips
< > SoC AC97 Audio support for SMDK6400 - WM9713
<*> SoC I2S Audio support for SMDK64X0 - WM8987
[*]   Proc-FS interface for audio codec control
< > SoC I2S Audio support for SMDK6400 - WM8990
< > SoC I2S Audio support for SMDK6400 - WM8990
< > SoC I2S Audio support for SMDK6400 - WM8753
< > SoC I2S Audio support for SMDK6400 - WM8990
```

测试：

录音。

/hhtech/app/record -o /tmp/out.wav

【注意】!

录音结束后，输入命令 quit退出程序，保存录音文件。可以加 -h参数得到 record程序的。

『说明』:

播放刚才的的录音：

/hhtech/app/mplayer /tmp/out.wav

播放 MP3: 启动开发板后，键入以下命令：

/hhtech/app/mp3player *.mp3

4.9 SD 卡

驱动位置：

/HHARM6410/s3c-linux-2.6.21/drivers/mmc/s3c-hsmmc.c

内核配置：

make menuconfig

进入 Device Drivers --->

MMC/SD Card support --->

选择

```
<*> MMC support
[ ]   MMC debugging
---   MMC block device driver
<*>   Samsung S3C High Speed MMC Interface support
[ ]     MMC can be used as a Root Filesystem Device
[*]     Fix IRQ Hanging in HS-MMC
[*]     Support Scatter and Gather in HS-MMC
---   Support moviNAND from Samsung Electronics
[ ] To debug interface, enable all debug messages
```

测试：

开发板启动后插上 sd 卡后终端显示：

```
# s3c-hsmmc: card inserted.
s3c-hsmmc: card inserted.
s3c-hsmmc: card removed.
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc0: new high speed SD card at address b368
mmcblk0: mmc0:b368          249856KiB
 mmcblk0: p1
yaffs: dev is 187695104 name is "mmcblk0"
yaffs: passed flags "flush,utf8"
yaffs: Bad mount option "flush"
yaffs: dev is 187695104 name is "mmcblk0"
yaffs: passed flags "flush,utf8"
yaffs: Bad mount option "flush"
```

现挂载 sd 卡

```
#mount /dev/mmcblk0p1 /mnt
#cd /mnt
#ls
lost+found
```

4.10 实时时钟（RTC）

驱动程序：
/HHARM 6410/s3c-linux-2.6.21/drivers/rtc/rtc-s3c.c
测试程序文件位置：busybox自带的 hwclock
华恒 HHARM 6410-Integration-R1带有 RTC，利用备用电池供电。
A. 读写 Linux 系统时钟

date //读取 linux系统时间
date 082216262008 //设置 Linux系统时间为 2008年 8月 22日，16点 26分。

B. RTC硬件时钟的操作：
键入以下命令即可读取 RTC的时间：

hwclock //读取 RTC硬件时钟
hwclock -w //把当前 Linux系统时间写入到 RTC硬件
hwclock -s //把 Linux系统时间同步成 RTC硬件时间

4.11 TV-OUT

驱动主文件：
/HHARM 6410/multimedia/TV_S3C6410/tvout_drv/
应用程序：
/HHARM 6410/multimedia/TV_S3C6410/tvout_app/
测试：
开发板起来后，加载驱动：
#**insmod /hhtech/modules/s3c_mfc.ko**
#**insmod /hhtech/modules/s3c-tvscaler.ko**
#**insmod /hhtech/modules/s3c-tvenc.ko**
运行测试程序：
#**/hhtech/app/tv_test 1 veggie.264**
就可以在 TV上看到硬件解码后的视频了。

『说明』：
1.测试程序 tv_test是把硬件解码的 h264数据通过 TV缩放和 TV编码输出到电视上，所以需要加载硬件编解码驱动 **s3c_mfc.ko**，针对一般硬件编解码的驱动模块也是 **s3c_mfc.ko**。如果是显示图片或把 mplayer软件解码的数据在 TV上输出就不需要加载这个驱动模块了。
2.harryp_320x240.264是提供的 h264编码的视频流；所有测试视频（包括 h263，h264，vc1和 mpeg4）在 /HHARM 6410/multimedia/test_media目录下

4.12 硬件编解码

硬件编解码部分只提供驱动模块，不提供源码，位于：

/HHARM6410/multimedia/FMV_MFC_V1.0/mfc_drv/

其中 s3c_mfc.ko是 TV输出的硬件编解码模块 mfc_for_tvout.ko与针对一般硬件编解码的驱动模块统一体。

应用程序位于：

/HHARM6410/multimedia/MFC/mfc_app/File_Operation/

测试：

加载驱动模块：

#**insmod /hhtech/modules/s3c_mfc.ko**

#**insmod /hhtech/modules/s3c_pp.ko**

运行测试程序：

#**/hhtech/app/mfc-display /mnt/veggie.264 h264**

就可以在 lcd上显示硬件解码后的视频流。

『说明』

1.MFC模块把视频文件解码为 YUV420格式，再通过 POST Processor进行颜色空间转换和缩放成为 RGB16 240x320的数据并在 LCD上显示。 s3c_pp.ko就是 post processor的驱动模块，驱动源文件位于 /HHARM6410/multimedia/post_processor/pp_drv目录下。

2.mfc-display后的两个参数，第一个参数表示需要解码的文件，第二个是文件格式，可以为 h264,mpeg4,h263 或 vc1。

3. /HHARM6410/multimedia/MFC/mfc_app/File_Operation/下的应用程序包括 mpeg4/h264/h263编码测试和 mpeg4/h264/h263/vc1解码测试以及在 lcd上显示。可以修改 test.c来编译不同的测试程序：

```
#include "line_buf_test.h"
#include "ring_buf_test.h"
#include "encoder_test.h"
#include "display_test.h"
#include "demo.h"
int main(int argc, char **argv)
{
        //Test_H263_Decoder_Line_Buffer(argc, argv);
        //Test_H264_Decoder_Line_Buffer(argc, argv);
        //Test_MPEG4_Decoder_Line_Buffer(argc, argv);
        //Test_Decoder_Ring_Buffer(argc, argv);
        //Test_Encoder(argc, argv);
        Test_Display(argc, argv);
        //Demo(argc, argv);
        return 0;
}
```

4.13 wifi

驱动内核自带：/drivers/net/wireless/libertas/

内核配置如下：

```
        Networking  --->
            Wireless  --->
```

```
<*> Generic IEEE 802.11 Networking Stack (mac80211)
[*]   'simple' rate control algorithm
[ ]   Enable debugging output
{*} Generic IEEE 802.11 Networking Stack
[ ]   Enable full debugging output
<*>   IEEE 802.11 WEP encryption (802.1x)
<*>   IEEE 802.11i CCMP support
<*>   IEEE 802.11i TKIP encryption
< >   Software MAC add-on to the IEEE 802.11 networking stack
```

```
Device Drivers  --->
     [*] Network device support  --->
        Wireless LAN  --->
```

```
[ ] Wireless LAN (pre-802.11)
[*] Wireless LAN (IEEE 802.11)
<M>   Marvell 8xxx Libertas WLAN driver support
< >     Marvell Libertas 8388 USB 802.11b/g cards
<M>     Marvell Libertas 8385 and 8686 SDIO 802.11b/g cards
[ ]     Enable full debugging output in the Libertas module.
< >   USB ZD1201 based Wireless device support
< >   Realtek 8187 USB support
< >   Softmac Prism54 support
```

然后在终端提示符下输入：

```
# modprobe libertas
# modprobe libertas_sdio
libertas_sdio: Libertas SDIO driver
libertas_sdio: Copyright Pierre Ossman
model=0xb
sd8686_helper.bin sd8686.bin
libertas: eth1: Marvell WLAN 802.11 adapter
# iwconfig
lo        no wireless extensions.
eth0      no wireless extensions.

eth1      IEEE 802.11b/g  ESSID:""
          Mode:Managed  Frequency:2.437 GHz  Access Point: Not-Associated
```

```
          Bit Rate:0 kb/s   Tx-Power=15 dBm
          Retry limit:8   RTS thr=2347 B   Fragment thr=2346 B
          Encryption key:off
          Power Management:off
          Link Quality:0  Signal level:0  Noise level:0
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

```
# iwlist scan
lo        Interface doesn't support scanning.
eth0      Interface doesn't support scanning.
eth1      Scan completed :
          Cell 01 - Address: 00:22:93:06:FC:E4
                    ESSID:"ChinaNet-TNT"
                    Mode:Managed
                    Frequency:2.412 GHz (Channel 1)
                    Quality=41/100  Signal level=-89 dBm  Noise level=-96 dBm
                    Encryption key:on
                    Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                              9 Mb/s; 12 Mb/s; 18 Mb/s; 24 Mb/s; 36 Mb/s
                              48 Mb/s; 54 Mb/s
                    IE: WPA Version 1
                        Group Cipher : TKIP
                        Pairwise Ciphers (1) : TKIP
                        Authentication Suites (1) : PSK
```

成功，若第一次搜索不到，可多搜几次。
iwconfig，iwlist在 /hhtech/app/下
libertas、 libertas_sdio 在 lib/modules/2.6.24.7/kernel/drivers/net/wireless/libertas/下，需要的话要断电重启，wifi模块的 reset引脚在软件中没有控制，所以直接 reset板子会导致wifi没有工作。

第五章 硬 件 系 统

5.1 功能模块结构图

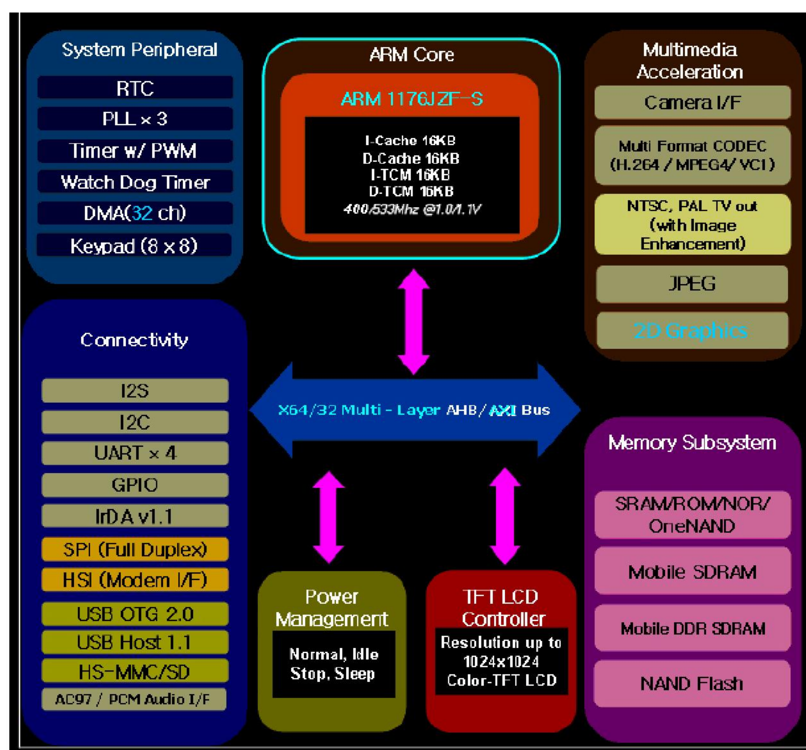


图 S3C6410功能模块）

5.2 各个部分的构成

存储部分：

1片 1G× 8 位数据宽度的 nand FLASH，共 1G 字节同时支持其他封装相同三星的 NAND，速度 150ns； 两片 32M× 16 位数据宽度的 DDRRAM 构成共 128M DDRRAM。

数据通讯口和外部输入输出口：

底板提供一 (或两)个四线 RS232 接口（COM 1）；一个 USB口（host），一个 LCD和触摸屏接口，一个 RTC接口，一个 MMCSD接口，一个 DM9000网口，两个音频接口（micphone，headphone），一路视频输入（saa7113，ov9650 二选一），一路视频输出（TV-out），一个 4X4 的键盘，一个 SDIO 接口 WIFI模块。其它接口请见底板的 PCB图。

5.3 片选

S3C6410提供 6路片选， nGCSn[0~ 6]，每个片选都指定了固定的地址，每个片选固定间隔为

128M 字节。具体参见 CPU手册 P5-2。

Address		Size(MB)	Description	Note
0x0000_0000	0x07FF_FFFF	128MB	Remap 0 : SRAM0 or Boot Loader Remap 1 : Internal ROM	Mirrored Region
0x0800_0000	0x0BFF_FFFF	64MB	Internal ROM	
0x0C00_0000	0x0FFF_FFFF	64MB	Stepping Stone (Boot Loader)	
0x1000_0000	0x17FF_FFFF	128MB	SMC Bank 0	
0x1800_0000	0x1FFF_FFFF	128MB	SMC Bank 1	
0x2000_0000	0x27FF_FFFF	128MB	SMC Bank 2	
0x2800_0000	0x2FFF_FFFF	128MB	SMC Bank 3	
0x3000_0000	0x37FF_FFFF	128MB	SMC Bank 4	
0x3800_0000	0x3FFF_FFFF	128MB	SMC Bank 5	
0x4000_0000	0x47FF_FFFF	128MB	Memory Port1 DDR/SDRAM Bank0	
0x4800_0000	0x4FFF_FFFF	128MB	Memory Port1 DDR/SDRAM Bank1	
0x5000_0000	0x5FFF_FFFF	256MB	Memory Port2 DDR/SDRAM Bank0	
0x6000_0000	0x6FFF_FFFF	256MB	Memory Port2 DDR/SDRAM Bank1	

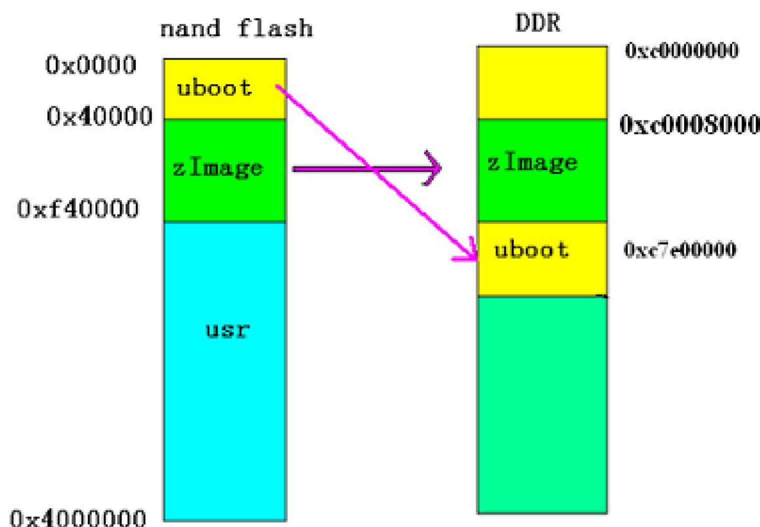
Note. Noting to be special.

HHARM 6410开发板内存由两片 32M × 16 位数据宽度的 DDR-RAM 构成，两片拼成 32位模式，公用 Memory Port2，共 128M RAM。起始地址：0x50000000。

NGCS1接的是一片 64M × 8 位数据宽度的 nand FLASH，安装在 SMC BANK2，起始地址为 0x0。其中内核 zImage烧写在地址 0x40000开始处。

u-boot的后部分代码是拷贝到 DDR的 0xc7e00000地址运行。

下面给出板上的地址空间分布：MEMORY MAP



5.4 中断

S3C6410可处理 64路中断，外加 28路外部中断。具体可参见 PDF手册 P325。

The S3C6400X supports 64 interrupt sources as shown in the table below.

Int. No.	Sources	Description	Group
63	INT_ADC	ADC EOC interrupt	TZIC1, VIC1
62	INT_PENDN	ADC Pen down interrupt	TZIC1, VIC1
61	INT_SEC	Security interrupt	TZIC1, VIC1
60	INT_RTC_ALARM	RTC alarm interrupt	TZIC1, VIC1
59	INT_IrDA	IrDA interrupt	TZIC1, VIC1
58	INT_OTG	USB OTG interrupt	TZIC1, VIC1
57	INT_HSMMC1	HSMMC1 interrupt	TZIC1, VIC1
56	INT_HSMMC0	HSMMC0 interrupt	TZIC1, VIC1
55	INT_HOSTIF	Host Interface interrupt	TZIC1, VIC1
54	INT_MSM	MSM modem I/F interrupt	TZIC1, VIC1
53	INT_EINT4	External interrupt Group 1 ~ Group 9	TZIC1, VIC1
52	INT_HSIrx	HSI Rx interrupt	TZIC1, VIC1
51	INT_HSItx	HSI Tx interrupt	TZIC1, VIC1
50	INT_I2C	I2C interrupt	TZIC1, VIC1
49	INT_SPI1/INT_HSMMC2	SPI1 interrupt or HSMMC2 interrupt	TZIC1, VIC1
48	INT_SPI0	SPI0 interrupt	TZIC1, VIC1
47	INT_UHOST	USB Host interrupt	TZIC1, VIC1
46	INT_CFC	CFCON interrupt	TZIC1, VIC1

板上扩展的外设接口占用片选、中断情况：

接口	占用片选	占用中断
DM9000	nGCS1	EINT10
NAND	NGCS2	

在开发板的终端可以通过如下命令查看板上的中断信息：

#**cat /proc/interrupts**

```
           CPU0
 22:          1     -  s3c-keypad
 26:          0     -  s3c2410-wdt
 28:    3386800     -  S3C Timer Tick
 30:          0     -  s3c-lcd
 37:       4498     -  s3c-uart
 50:         44     -  s3c2410-i2c
 62:        518     -  s3c_updown
 63:       7170     -  s3c_action
 74:      84739     -  eth0
Err:          0
```

关于中断向量号的定义，可参见：

kernel/include/asm/arch/irqs.h

```
/* EINT 0 ~27 */
#define IRQ_EINT0    S3C6410_IRQ(64)
#define IRQ_EINT1    S3C6410_IRQ(65)
#define IRQ_EINT2    S3C6410_IRQ(66)
#define IRQ_EINT3    S3C6410_IRQ(67)
#define IRQ_EINT4    S3C6410_IRQ(68)
#define IRQ_EINT5    S3C6410_IRQ(69)
#define IRQ_EINT6    S3C6410_IRQ(70)
```

```
#define IRQ_EINT7    S3C6410_IRQ (71)
#define IRQ_EINT8    S3C6410_IRQ (72)
#define IRQ_EINT9    S3C6410_IRQ (73)
#define IRQ_EINT10   S3C6410_IRQ (74)
#define IRQ_EINT11   S3C6410_IRQ (75)
#define IRQ_EINT12   S3C6410_IRQ (76)
#define IRQ_EINT13   S3C6410_IRQ (77)
#define IRQ_EINT14   S3C6410_IRQ (78)
#define IRQ_EINT15   S3C6410_IRQ (79)
#define IRQ_EINT16   S3C6410_IRQ (80)
#define IRQ_EINT17   S3C6410_IRQ (81)
#define IRQ_EINT18   S3C6410_IRQ (82)
#define IRQ_EINT19   S3C6410_IRQ (83)
#define IRQ_EINT20   S3C6410_IRQ (84)
#define IRQ_EINT21   S3C6410_IRQ (85)
#define IRQ_EINT22   S3C6410_IRQ (86)
#define IRQ_EINT23   S3C6410_IRQ (87)
#define IRQ_EINT24   S3C6410_IRQ (88)
#define IRQ_EINT25   S3C6410_IRQ (89)
#define IRQ_EINT26   S3C6410_IRQ (90)
#define IRQ_EINT27   S3C6410_IRQ (91)
```

5.5 GPIO

S3C6410提供 188 路复用的 IO口线，分为如下 17组端口进行管理：

— PortA (GPA): 8路输出口线
— PortB (GPB): 7路输入 输出口线
— PortC (GPC): 8路输入 输出口线
— PortD (GPD): 5路输入 输出口线
— PortE (GPE): 5路输入 输出口线
— PortF (GPF): 16路输入 输出口线
— PortG (GPG): 7路输入 输出口线
— PortH (GPH): 10路输入 输出口线
— Port I (PG I): 16路输入 输出口线
— PortJ (PG J): 12路输入 输出口线
— PortK (PGK): 16路输入 输出口线
— PortL (PGL): 15路输入 输出口线
— PortM (PGM): 6路输入 输出口线
— PortN (PGN): 16路输入 输出口线
— PortO (PGO): 16路输入 输出口线
— PortP (PGP): 15路输入 输出口线
— PortQ (PGQ): 9路输入 输出口线

核心板上将这些复用的信号引脚中未被占用的全部引到底板上来，具体请参见产品光盘中

的 PROTEL格式的电路原理图和 PCB。

5.6 总线

1. S3C6410 是地址总线宽度 32，数据总线宽度 32位。533M 的主频，133M 的总线速度。
2. HHARM 6410的数据总线宽度是可配置的，可分别配为 16 位或 8 位模式。若外接 8 位或 16 位数据宽度的外设芯片，与 CPU 相接时，设置是在 SROM_BW（参见 S3C6410X User's Manual 的 SROM Controller）实现的。在给外设分配片选时，设置好它的 BWSCON 中的 datawidth位，在访问它的地址时就可以改变数据宽度。

16 位数据宽度时，是低 16 位数据线有效；

8 位模式时，是最低 8 位数据线有效。

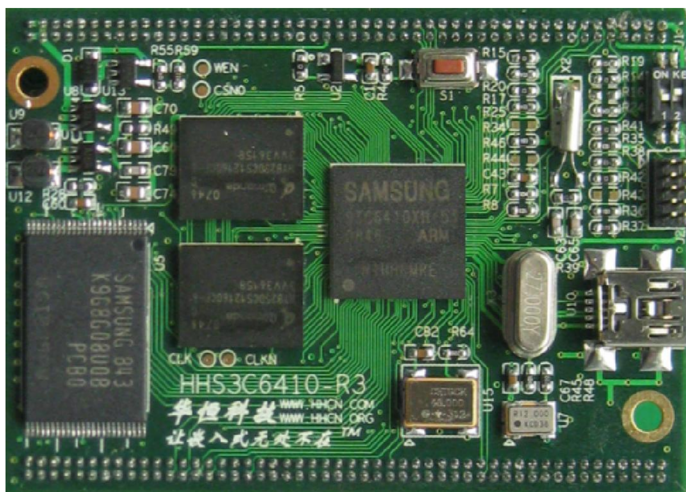
【注意】！

启动时这个对 CS0 是无效的，因为 CS0 是接存放启动代码的存贮器片选，一般都是 FLASH，在 CPU 刚加电时，这时的数据宽度就无法用 BWSCON 来设置了，就只有硬件实现了：由复位后硬件配置决定数据的宽度，复位默认为 0x00000000。

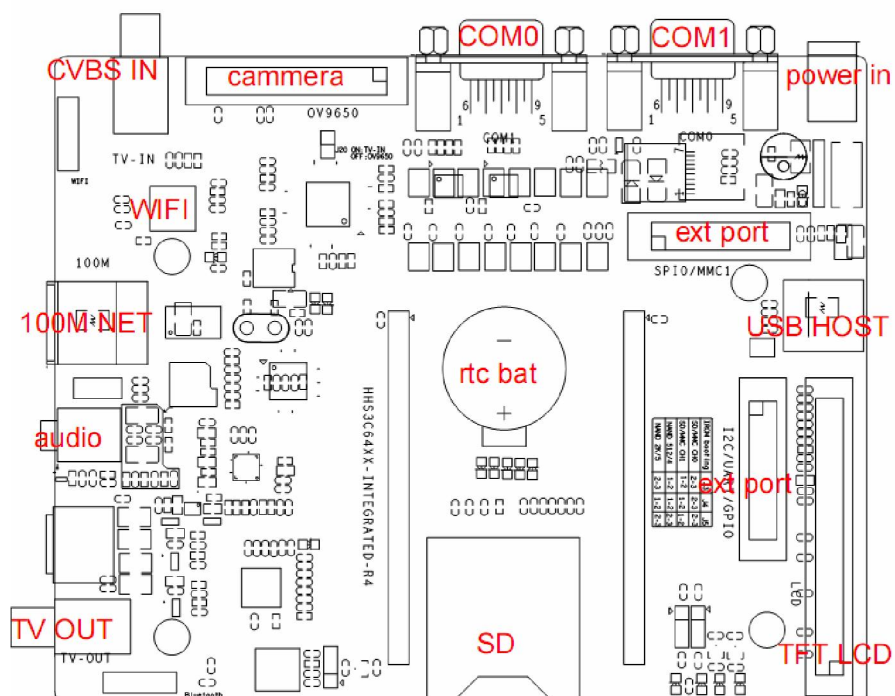
两片 SDRAM 为 32 位寻址，但两片的数据总线分别接 HHARM 6410高 16 位和低 16位数据总线，这样拼成 32 位 SDRAM 使用，所以两片 SDRAM 共享一个 CS。而一片 FLASH 则固定为 16 位数据读写访问模式，它就只接 HHARM 6410的低 16 位数据总线。

5.7 外设接口图

核心板正面俯视图：


SAMSUNG
HHS3C6410-R3
华恒科技
WWW.HHCN.COM
WWW.HHCN.ORG
让嵌入式无处不在

外设接口底板图：



HHARM 6410型基本底板的主要接口和器件排布参见上图。

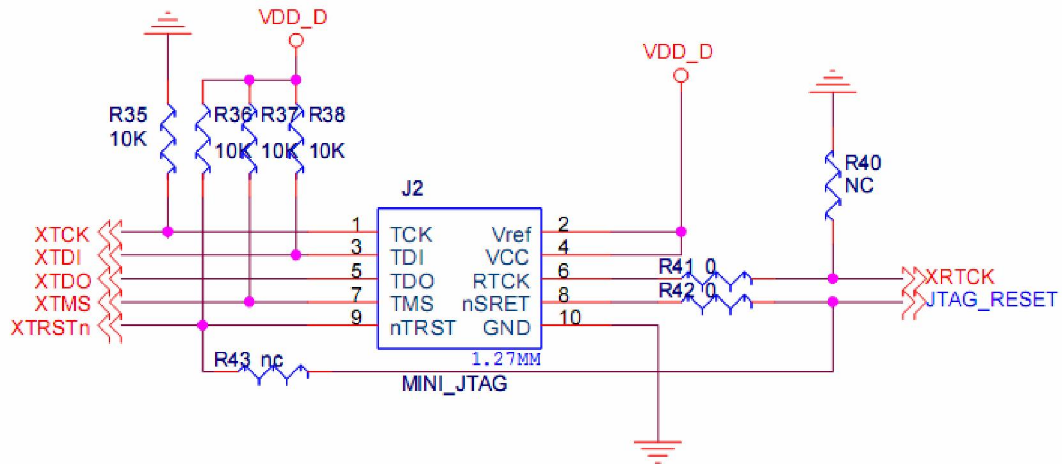
5.8 接口管脚说明

1.LED 指示灯

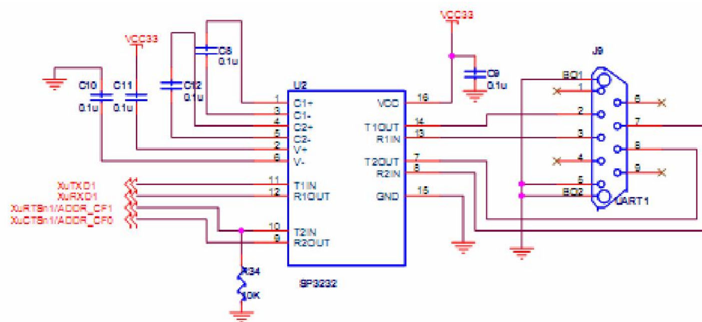
底板（基本板）：

状态指示部分：由 LED 提供电源指示。

2.JTAG 接口



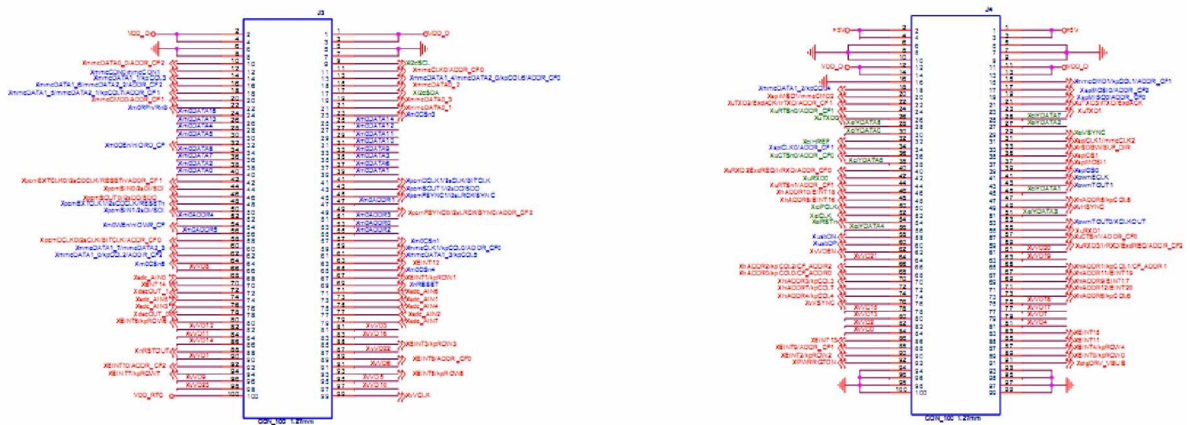
3.RS232 COM 接口：



管脚说明：

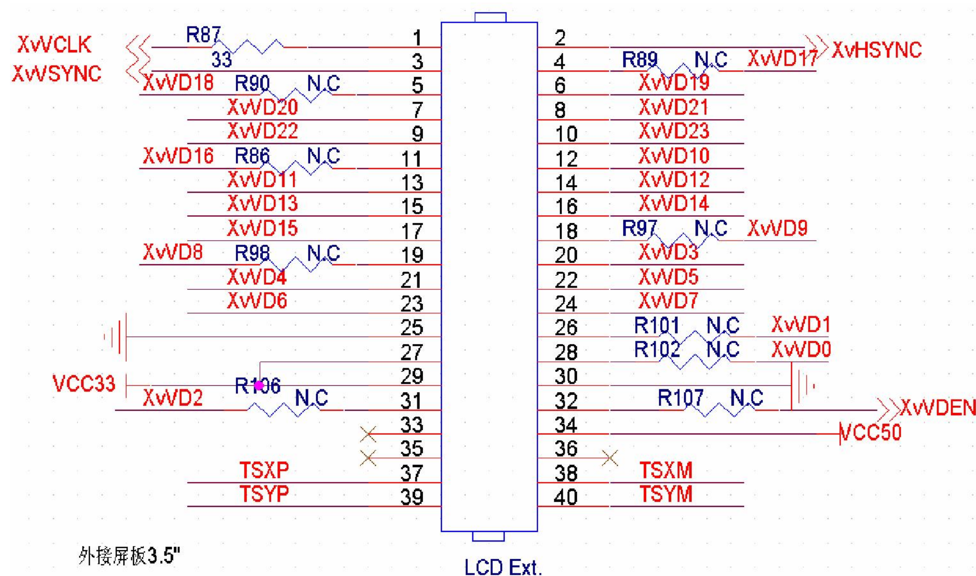
TX1	串行口 1 发送线
RX1	串行口 1 接受线

4接口引脚图



详细情况请见光盘上原理图。

5. LCD和触摸屏接口



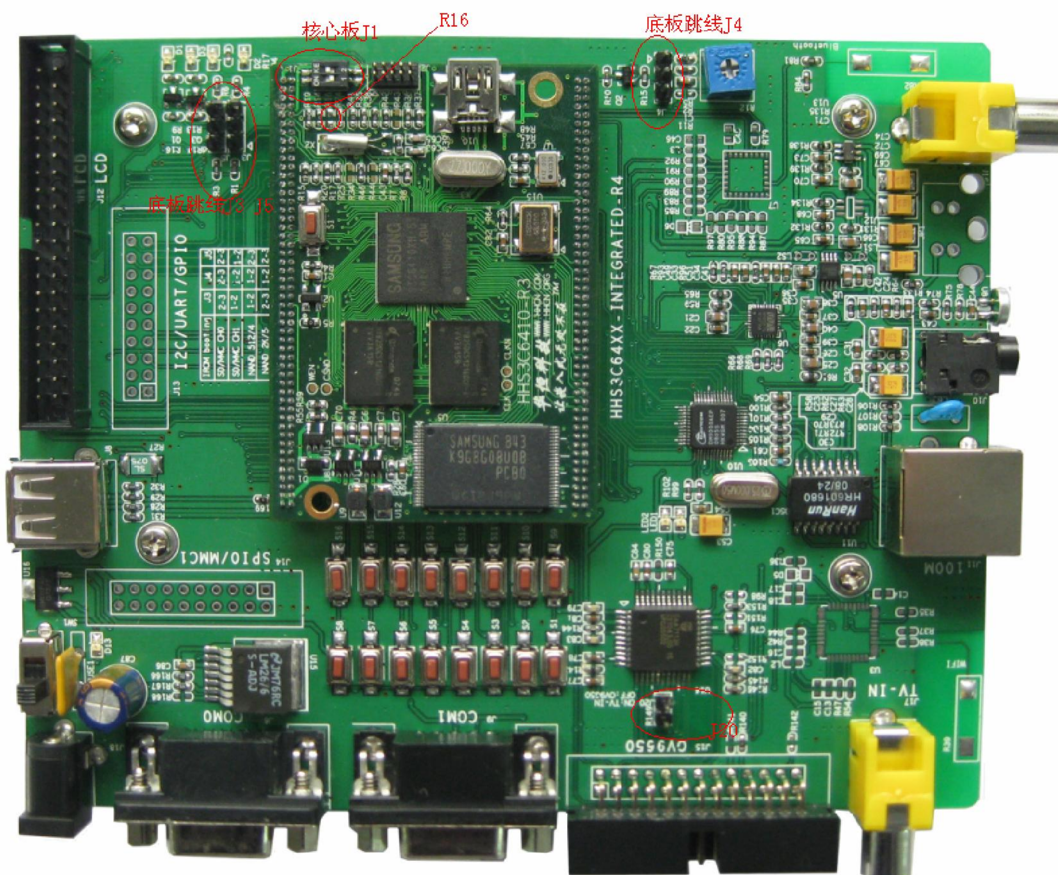
6. 电气特性

电源	+9V (0 ~ +10%)
工作环境温度 (注 1)	0 C ~ 40 C
保藏温度	－20 C ~ 70 C
相对湿度	25% ~95%

注 1：可以提供满足 -40 C ~ 50 C工作环境温度的产品；

5.9 跳线及启动方式说明

跳线说明



1.底板跳线 J20 接上跳线帽选择从 TV-IN接口输入视频数据，去掉选择从 OV9650输入。

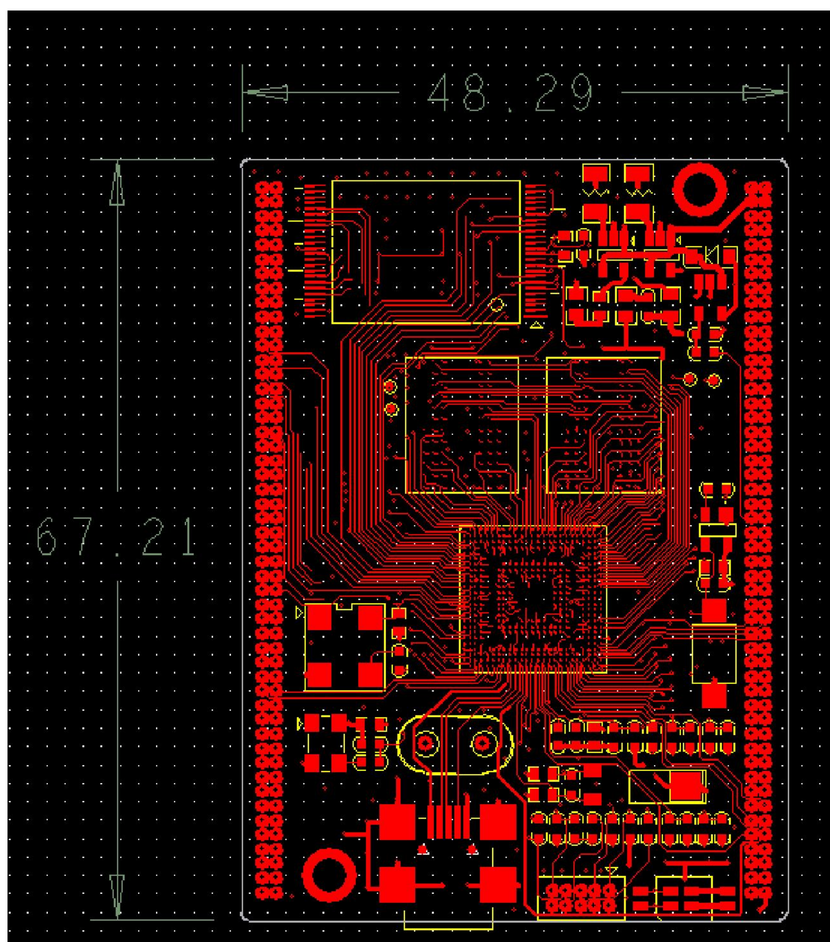
2.底板跳线 J3,J4,J5 配合核心板 J1选择启动模式，为了尽量确保固件不被误操作删掉同时使用一个电阻来确定启动方式。

启动方式	R16	J1	J3	J4	J5
Nand 512/4	NC(不焊)	1OFF;2OFF	X	X	X
Nand 2K/5	NC	1OFF;2ON	X	X	X
Nor flash	NC	1ON;2OFF	X	X	X
MMC0 *	0R(0欧姆)	1ON;2ON	2-3	2-3	2-3

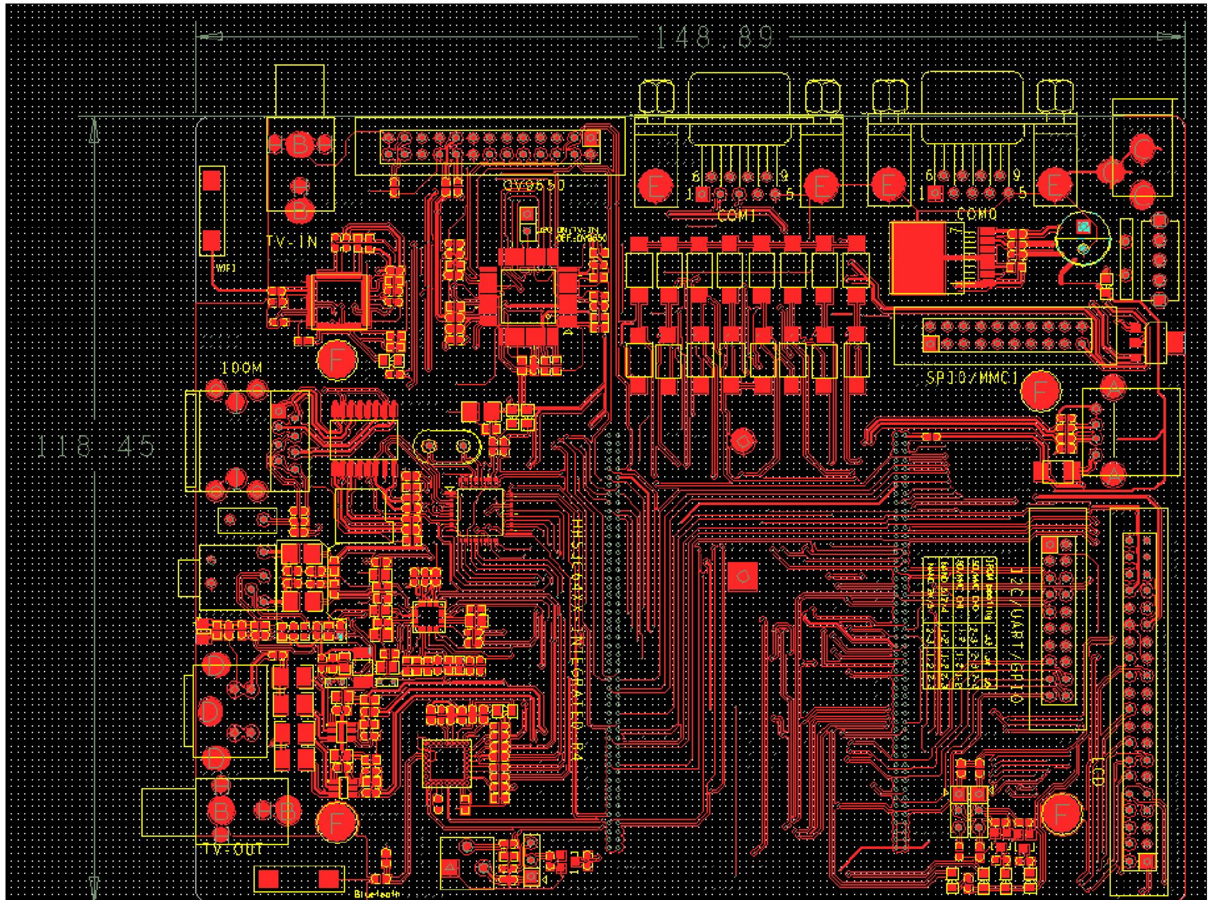
出厂时默认启动方式 Nand 2K/5;

第六章 机 械 特 性

核心板



底板



第七章 底 板 的 硬 件 设 计

7.1 基本板的设计

基本板包括如下主要部分：

与核心板的连接器；

基本端口：包括两个四线串口 COM0, 1；

一个触摸屏；

一个 10M/100M 网口；

一个 IIS 音频接口等。

电源：将外部输入的未稳压电源稳定为核心板和基本板上芯片所需的 3.3V，5V

对于串行口 1因为在不同的应用中可能会被作为 RS232、RS485/422、红外接口使用，因此对该口必须使用相应的电平转换芯片。在基本板的设计中华恒科技给出的是作为 RS232 端口的方案。

7.2 用户底板原理性设计和硬件方案制定

7.2.1 基本端口的设计

以串口、LCD、触摸屏的设计可以直接参考基本板的设计，具体电路请参考基本板的原理图和PCB。另外必须注意的是，在串行口电平转换芯片的选择和使用中必须注意电平兼容问题，在基本板中华恒科技使用的是 3.3V 工作电压的 RS232 电平转换芯片，2.5V工作电压的 Lanswitch芯片。

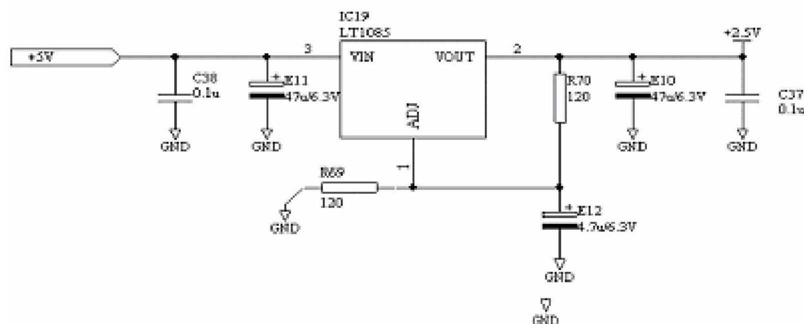
7.2.2 电源的设计

核心板工作电源为 3.3V，1.0－1.2V,1.8V,2.5V直流，在基本板中，为了得到更高的电源使用效率华恒科技都采用 DC/DC提供电源电压。

在其他的应用设计中根据不同的电源消耗需求，可以选择线性稳压源方案和开关稳压源方案。对于前一种选择，可以获得低噪声、廉价等益处，但同时也有效率低，发热较大等缺点；对于开关电源方案正好与线性电源的优缺点相反。

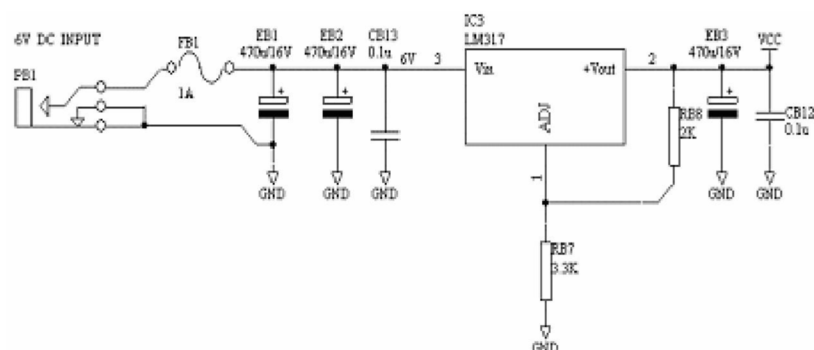
使用线性电源时，华恒科技给出以下两种参考：

其一参见下图：



这是一个使用低压差线性稳压器获得 2.5V 稳压输出的例子，输出电压的计算可以使用以下的公式：Vo=1.25V * (R69+R70)/R70；芯片上的耗散功率可以使用以下公式计算：Pc=(Vi-Vo)*Io，由上可以看到，除了输出电流，输入和输出间的压差正比于芯片上的耗散功率。稳压芯片的金属背板表面工作温度必须小于 80 摄氏度，设其金属背板到散热器的热阻为 Rt1，散热器到空气的热阻为 Rt2，则稳压芯片金属背板与环境的温升为：T=Pc/(Rt1+Rt2);由此可以看到减少温升必须减小 Pc 或者总热阻，后者可以通过减小 Rt1（使用导热硅脂于散热器于管壳之间），减少 Rt2（增大散热器面积）；

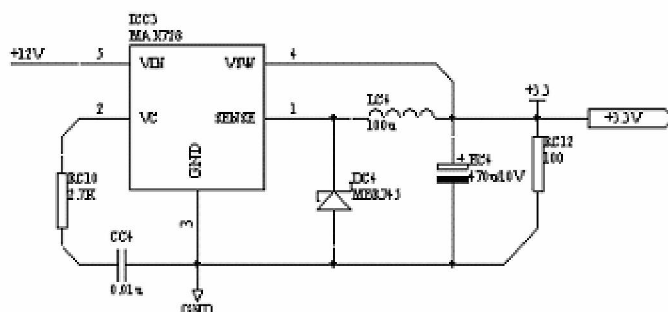
其二参见下图：



这是一个使用最廉价的普通线性稳压器获得 3.3V 稳压输出的例子，输出电压的计算可以使用以下的公式：Vo=1.25V * (RB8+RB7)/RB8；其他同上，但要注意的是 LM317 必须有大于 2.5V 的输入输出压差才能工作在稳压状态。因此最下输入电压要大于上者。

使用开关电源时，华恒科技给出以下参考：

见下图：



要注意的是电感 LC4 的选择，其最小平均直流工作电流必须大于输出电流（既在规定的输出电流下其不能出现磁芯饱和现象）。另外必须注意 PCB的排版，这将决定电源的噪声幅度，在以上电路中 EC4 为钽电解电容，在一般的应用中必须另外附加并联一个 0.1u 的瓷介电容 以滤除高频分量。电源类型的选择，最终是效率、成本、体积等因素的综合考虑。

7.2.3 其它电路部分的设计

在这里特别指出应当充分重视对核心板各个引出线添加缓冲电路，以充分保证核心板和处理器的稳定性；对于核心板连接器的地线和电源线引脚应当全部连接到相应的地和电源上。

7.2.4 PCB 设计和排版时的注意事项

PCB 布局时的考虑

首先由于 100PIN 双列脚比较密，无法在焊盘间穿线，在核心板设计中将上述插座的外围引脚大部分作为电源或地线。其次由于核心板的读写访问速率较高，因此必须考虑到布线传输延迟对电路的影响。在底板 PCB 走线设计中时应当将涉及到数据总线、地址总线、内存访问控制信号和中断信号等的电路部分排布在 CN1,CN2 连接器之间，以方便布线和减少信号延迟。简单应用时可以使用 2 层板，并且大面积铺地；复杂的应用使用多层布线，并且至少有一层为地层（最好将相邻的层作为电源层。当板上有数个电源品种时，一般来讲如果可以将使用同种电源的电路部分布局在临近区域内，然后将电源层分割为数个区块。）

另外还应当注意以下事项：

核心板接口插座的整体布局；

板上提供必要的测试焊盘；

7.2.5 PCB 在电路稳定性和抗干扰方面的考虑

应重点考虑以下几点：

1.合理的地层和电源层分割

建议使用较为完整的一个 PLANE 作为地层，如果存在多个“地”（例如：数字地，模拟地，保护地等）可以在同一个地层上适当分割获得。对于电源层类似上述，并且应当注意低噪声电路部分电源和高噪声电路部分电源的隔离或滤波。

2.合理的去耦电容排布

在电源进线端，在各个数字芯片电源引脚附近等分布适当的去耦电容。对核心板的隔离，以及在各个端口使用适当的输入输出缓冲；

附 录

附录 A LINUX 常见术语

Linux

"Linux 是一种 UNIX 操作系统的克隆，它（的内核）由 Linus Torvalds 以及网络上组织松散的黑客队伍一起从零开始编写而成。Linux 的目标是保持和 POSIX 的兼容。

"Linux 具备现代一切功能完整的 UNIX 系统所具备的全部特征，其中包括真正的多任务、虚拟内存、共享库、需求装载、共享的写时复制程序执行、优秀的内存管理以及 TCP/IP 网络支持等。

"Linux 的发行遵守 GNU 的通用公共许可证。

"Linux 起初为基于 386/486 的 PC 机开发，但现在，Linux 也可以运行在 DEC Alpha、SUN Sparc、M68000，以及 MIPS 和 PowerPC 等计算机上。"

POSIX

POSIX 表示可移植操作系统接口（Portable Operating System Interface，缩写为 POSIX 是为了读音更像 UNIX）。电气和电子工程师协会（Institute of Electrical and Electronics Engineers，IEEE）最初开发 POSIX 标准，是为了提高 UNIX 环境下应用程序的可移植性。然而，POSIX 并不局限于 UNIX。许多其它的操作系统，例如 DEC OpenVMS 和 Microsoft Windows NT，都支持 POSIX 标准，尤其是 IEEE Std. 1003.1-1990（1995 年修订）或 POSIX.1，POSIX.1 提供了源代码级别的 C 语言应用编程接口（API）给操作系统的服务程序，例如读写文件。POSIX.1 已经被国际标准化组织（International Standards Organization，ISO）所接受，被命名为 ISO/IEC 9945-1:1990 标准。

POSIX 现在已经发展成为一个非常庞大的标准族，某些部分正处在开发过程中。POSIX 与 IEEE 1003 和 2003 家族的标准是可互换的。

GNU

GNU 是 GNU Is Not UNIX 的递归缩写，是自由软件基金会的一个项目，该项目的目标是开发一个自由的 UNIX 版本，这一 UNIX 版本称为 HURD。尽管 HURD 尚未完成，但 GNU 项目已经开发了许多高质量的编程工具，包括 emacs 编辑器、著名的 GNU C 和 C++ 编译器（gcc 和 g++），这些编译器可以在任何计算机系统上运行。所有的 GNU 软件和派生工作均适用 GNU 通用公共许可证，即 GPL。GPL 允许软件作者拥有软件版权，但授予其他任何人以合法复制、发行和修改软件的权利。

Linux 的开发使用了许多 GNU 工具。Linux 系统上用于实现 POSIX.2 标准的工具几乎都是 GNU 项目开发的，Linux 内核、GNU 工具以及其他一些自由软件组成了人们常说的 Linux：

符合 POSIX 标准的操作系统 Shell 和外围工具。

C 语言编译器和其他开发工具及函数库。

X Window 窗口系统。

各种应用软件，包括字处理软件、图象处理软件等。

其他各种 Internet 软件，包括 FTP 服务器、WWW 服务器等。

关系数据库管理系统等。

GPL

GPL (General Public License)文本保存在 Linux 系统的不同目录下的命名为 COPYING 的文件里。例如，键入 cd /usr/doc/ghostscript* 然后再键入 more COPYING 可查看 GPL 的内容。

GPL 和软件是否免费无关，它主要目标是保证软件对所有的用户来说是自由的。GPL 通过如下途径实现这一目标：

它要求软件以源代码的形式发布，并规定任何用户能够以源代码的形式将软件复制或发布给别的用户。

它提醒每个用户，对于该软件不提供任何形式的担保。如果用户的软件使用了受 GPL 保护的任何软件的一部分，那么该软件就继承了 GPL 软件，并因此而成为 GPL 软件，也就是说必须随应用程序一起发布源代码。

GPL 并不排斥对自由软件进行商业性质的包装和发行，也不限制在自由软件的基础上打包发行其他非自由软件。遵照 GPL 的软件并不是可以任意传播的，这些软件通常都有正式的版权，GPL 在发布软件或者复制软件时声明限制条件。但是，从用户的角度考虑，这些根本不能算是限制条件，相反用户只会从中受益，因为用户可以确保获得源代码。

尽管 Linux 内核也属于 GPL 范畴，但 GPL 并不适用于通过系统调用而使用内核服务的应用程序，通常把这种应用程序看作是内核的正常使用。

Linux 的主要发行版

Ubuntu Linux采用 deb的软件保管理方式，软件的安装、卸载和升级非常方便，并提供了大量的图形化管理工具，是初学者的最佳选择。

Mandrake

Slackware

Debian GNU/Linux

是由 GNU 发行的 Linux 版本，最符合 GNU 精神。提供了最大的灵活性， 适合 Linux 的高级用户。

附录 B 常用 LINUX 命令

以下均以 Ubuntu Linux为例说明。

基本命令：

ls：显示当前目录下的所有文件和目录。

ls – a ：可以看到隐藏的文件，如以 开头的文件。

pwd：显示当前目录路径。

ps：列举当前 TTY 下所有进程

ps – A：列举所有

cd 目录名：进入目录

mkdir 目录名：创建目录

rmdir 目录名：删除空目录

rm – rf 目录名：强行删除整个目录内容（无法恢复），其中 f表示强制不进行提示， r表示目录递归。

Linux下的文件和目录是区分大小写的。

TAB文件目录匹配搜索的使用

例如华恒软件安装的目录为：/HHARM6410，假设 /目录下没有其它以 HHA 字符开头的其它目录和文件，则要进入这个目录，只需敲入：

cd /HHA

然后按下 TAB键，则 SHELL会自动匹配找到 /HHARM6410目录，这样就不必完全键入剩余的 RM6410字符，这个功能在访问名字很长的文件和目录时非常有效，可以大大提供键盘输入的速度，极为方便。

ncftp工具的使用：

ncftp是 Linux下非常好的 FTP工具软件，它除了支持 FTP命令操作外，还支持 Linux SHELL下的命令用法，例如，它也支持 TAB键补全，支持目录上传和下载（用 -r 或 -R参数）。ncftp 的用法，例如要 FTP一台 IP为 192.168.2.111的 Linux PC机 A，命令如下：

ncftp –u hhcn 192.168.2.111

其中 hhcn为 A机器上的合法的用户，连接上之后会提示输入 hhcn用户的密码，密码验证通过后，就进入 ncftp命令提示符。

编程时获取帮助 man (类似于 VC编程中的 MSDN)

man，即 manunal，是 UNIX 系统手册的电子版本。根据习惯，UNIX 系统手册通常分为不同的部分（或小节，即 section），每个小节阐述不同的系统内容。目前的小节划分如下：

命令：普通用户命令

系统调用：内核接口

函数库调用：普通函数库中的函数

特殊文件：/dev 目录中的特殊文件

文件格式和约定：/etc/passwd 等文件的格式

游戏。

杂项和约定：标准文件系统布局、手册页结构等杂项内容

系统管理命令。

内核例程：非标准的手册小节。

手册页一般保存在 /usr/man 目录下，其中每个子目录（如 man1,man2, ...,manl,mann）包含不同的手册小节。使用 man 命令查看手册页。

常用 man命令行：

man strtoul

取消 root密码：

vim /etc/shadow

可以看到第一行内容大致如下：

root:1dVVd5YVP$OgZG58TL/NRExTfcr6URH.:11829:0:99999:7:-1:-1:134539236

要取消 root密码，只需将第一行 root后第一对 之间的字符全部删除即可，删除后如下：

root::11829:0:99999:7:-1:-1:134539236

然后用 w强行存盘（因为 shadow文件是只读的）后用 q退出 vi则实现取消了 root密码。

修改 PC机 IP地址：

ifconfig eth0 192.168.2.111

压缩 解压缩：

Linux的压缩文件一般是以 gz或 .tar或者 .targz结尾的。前者是由 gzip压缩的，后者是先用 tar 归档，在用 gzip压缩而成的。

1. 以 gz结尾的为压缩文件，用命令： gzip -d filename来解压，得到的文件在当前目录中，但已没有了 gz。

2. 以 .tar结尾的为归档文件，用命令： tar -xvf filename来展开，生成的文件与源文件在同一目录中，只是少了 .tar。

3. 以 .targz结尾的文件最常见，可直接用命令： gzip -cd filename | tar xfv 来解开。

tar的用法：

解压： x参数表示解压

tar xzf hharm6410.tgz

把一个目录 HHARM6410压缩成一个文件： hharm6410.tgz

tar czf hharm6410.tgz HHARM6410

c参数表示压缩。

查找文件，如：main.c:

find –name main.c

或者：

locate shadow

【注意】！ locate为模糊匹配，它会递归的在当前目录下的所有子目录下搜索，并列出所有名字包含 shadow字串的文件。

在一个目录下（含子目录）的所有文件中查找含有某个字符串（如 " Modified by hhcn"）的所有文件：

grep 'Modified by hhcn' * -r

ctags的用法：

使用' ctags – R *'命令先在当前目录下建立一个 tags文件,那么只要在 源代码目录下由 vim 去开启文件的话，会自动加载 tags 这个文件，无需另行加载，否则要由 :set tags=your.tags 来指定 tags 文件。然后就是照一般使用 Vim ，光标移到标识符或函数名上，按 Ctrl+], 要回到原处就按

Ctrl+T。

请注意，Vim 激活时，工作目录（vim 激活时的所在目录）名为 tags 的文件会自动加载，$VIMRUNTIME/doc 及 $HOME/.vim/doc

目录下的 tags 文件也会自动加载。而且，凡是加载的 tags 文件里头所有标志文字都可以使用补全键来补全，别忘了这个好用的功能。

vi(m)用法

vi是 Linux/Unix 世界里极为普遍的全屏幕文本编辑器，几乎可以

说任何一台 Linux/Unix 机器都会提供这个软件。

vi有三种状态，即编辑方式、插入方式和命令方式。

在命令方式下，所有命令都要以 开始，所键入的字符系统均作命令

来处理，如 :q代表退出， :w表示存盘。

当你进入 vi时，会首先进入命令方式（同时也是编辑方式）。按下 I

就进入插入方式，用户输入的可视字符都添加到文件中，显示在屏幕上。按下 ESC就可以回到命令状态（同时也是编辑方式）。

编辑方式和命令方式类似，都是要输入命令，但它的命令不要以 开

始，它直接接受键盘输入的单字符或组合字符命令，例如直接按下 u就表示取消上一次对文件的修改，相当于 WINDOWS下的 Undo操作。编译方式下有一些命令是要以 开始的，例如查找字符串就是： /string则在文件中匹配查找 string字符串。在编辑模式下按下 就进入命令方式。

基本命令解释：

1. 光标命令

k、j、h、l——上、下、左、右光标移动命令。虽然您可以在 Linux中使用键盘右边的 4个光标键，但是记住这 4个命令还是非常有用的。这 4个键正是右手在键盘上放置的基本位置。

nG——跳转命令。 n为行数，该命令立即使光标跳到指定行。

Ctrl+G——光标所在位置的行数和列数报告。

w、 b——使光标向前或向后跳过一个单词。

2. 编辑命令

i、a、r——在光标的前、后以及所在处插入字符命令 (i=insert、a=append、r=replace)。

cw、 dw——改变 (置换)删除光标所在处的单词的命令 (c=change、d=delete)。

x、 d$、 dd——删除一个字符、删除光标所在处到行尾的所有字符以及删除整行的命令。

3. 查找命令

---- /string、 ?string——从光标所在处向后或向前查找相应的字符串的命令。

4. 拷贝复制命令

---- yy、 p——拷贝一行到剪贴板或取出剪贴板中内容的命令。

常用操作：

无论是开启新档或修改旧文件，都可以使用 vi，所需指令为：

$ vi filemane

如果文件是新的，就会在荧幕底部看到一个信息，告诉用户正在创建新文件。如果文件

早已存在，vi则会显示文件的首廿四行，用户可再用光标（cursor）上下移动。

~

~

上面是一个经 vi开启的模拟文件，一行开始处的波折号（~）表示文件的结尾。

—指令 i在光标处插入正文

—指令 I在一行开始处插入正文

—指令 a在光标後追加正文

—指令 A在行尾追加正文

—指令 o在光标下面新开一行

—指令 O在光标上面新开一行

在插入方式下，不能打入指令，必需先按〈Esc〉键，返回命令方式。假若户不知身处何态，也可以按〈Esc〉键，不管处於何态，都会返回命令方式。

在修改文件时，如何存档及退出指定文件都非常重要。在 vi内，行使存档或退出的指令时，要先按冒号（:），改变为命令方式，用户就可以看见在荧幕左下方，出现冒号（:），显示 vi已经改为指令态，可以进行存档或退出等工作。

:q!放弃任何改动而退出 vi，也就是强行退出

:w存档

:w!对于只读文件强行存档

:wq存档并退出 vi

:x与 wq的工作一样

:zz与 wq的工作一样删除正文

删除或修改正文都是利用编辑方式，故此，下面所提及的指令只需在编辑方式下，直接键入指令即行。

— x删除光标处字符（Character）

— nx删除光标处後 n个字符

— nX删除光标处前 n个字符

— ndw删除光标处下 n个单词（word）

— dd删除整行

— d$或 D删除由光标至该行最末

— u恢复前一次所做的删除

当使用 vi修改正文，加减字符时，就会采用另一组在编辑方式下操作的指令。

— r char由 char代替光标处的字符

— R text〈Esc〉由 text代替光标处的字符

— cw text〈Esc〉由 text取代光标处的单词

— C text〈Esc〉由 text取代光标处至该行结尾处

— cc使整行空白，但保留光标位置，让你开始打入

—如删除指令一样，在指令前打入的数，表示执行该指令多少次。

要检索文件，必需在编辑方式下进行。

—／str〈Return〉向前搜寻 str直至文件结尾处

— ?str〈Return〉往後搜寻 str直至文件开首处

— n同一方向上重复检索

—N相反方向上重复检索
—vi缠绕整个文件，不断检索，直至找到与模式相匹配的下一个出现。

全程替换命令：
%s/string1/string2/g 在整个文件中替换"string1"成"string2"。
如果要替换文件中的路径：
使用命令"%s#/usr/bin#/bin#g"可以把文件中所有路径/usr/bin换成/bin。也可以使用命令"%s/\/usr\/bin/\/bin/g"实现，其中"\"是转义字符，表明其后的"/"字符是具有实际意义的字符，不是分隔符。

同时编辑2个文件，拷贝一个文件中的文本并粘贴到另一个文件中：
命令如下：
----vi file1 file2
----yy 在文件1的光标处拷贝所在行
----:n 切换到文件2 (n=next) 或者按 ctrl+ww，就在两个文件间切换。
----p 在文件2的光标所在处粘贴所拷贝的行
----:n 切换回文件1

将文件中的某一部分修改保存到临时文件，例如仅仅把第20~59行之间的内容存盘成文件/tmp/1，可以键入如下命令。
----vi file
----:20,59w /tmp/1

如果要在vi执行期间，转到shell执行，使用惊叹号（!）执行系统指令，例如在vi期间，列出当前目录内容，可以键入：
:!ls
另一方面，用户可以在主目录中创建.exrc环境文件，用set打入选项，每次调用vi时，就会读入.exrc中的指令与设置。下面是.exrc环境文件的实例：
set wrapmarging=8
set showmode
set autoindent
minicom用法
minicom是一个常用软件，它使用配置文件/etc/minirc.dfl，华恒光盘安装时会提供这个文件。
【注意】！
minicom占用串口，能且仅能启动一个minicom，启动第二个时就会报错：Device /dev/modem is locked. 其中/dev/modem就是/dev/ttyS0，即PC机串口1，它是在光盘安装时执行./arminst时创建的链接。查看arminst文件，可以看到如下一行：
ln -sf /dev/ttyS0 /dev/modem
minicom所有的操作都以ctrl+A开始，例如：退出为ctrl+A，松手后再按下Q，则弹出如下一个小框：选Yes即可退出minicom。

```
Leave without reset?
   Yes        No
```

minicom中最重要的操作就是对其进行配置的修改。这个操作要先 ctrl+A，松手后按下 O，则弹出如下框：

```
-[configuration]-
 Filenames and paths
 File transfer protocols
 Serial port setup
 Modem and dialing
 Screen and keyboard
 Save setup as dfl
 Save setup as..
 Exit
```

选择第三项 “Serial port setup”，则弹出下面框：

```
A -     Serial Device       : /dev/modem
B - Lockfile Location       : /var/lock
C -    Callin Program       :
D -   Callout Program       :
E -     Bps/Par/Bits        :115200 8N1
F - Hardware Flow Control   : No
G - Software Flow Control   : No

   Change which setting?
```

键入 E则弹出如下框，可改变波特率：

```
-[Comm Parameters]-

Current:115200 8N1

  Speed            Parity          Data

A: 300             L: None         S: 5
B: 1200            M: Even         T: 6
C: 2400            N: Odd          U: 7
D: 4800            O: Mark         V: 8
E: 9600            P: Space
F: 19200
G: 38400
H: 57600
I: 115200          Q: 8-N-1
J: 230400          R: 7-E-1


Choice, or <Enter> to exit?
```

若要使用 PC机的串口 2来接开发板的串口 1做监控,则要在串口配置框中选择 A,即 “Serial Device”，则原来的配置框第一行进入编辑模式，将原来的 /dev/modem改为如下的： /dev/ttyS1，即串口 2。

```
A -     Serial Device        : /dev/ttyS1
B - Lockfile Location        : /var/lock
C -    Callin Program        :
D -   Callout Program        :
E -     Bps/Par/Bits         :115200 8N1
F - Hardware Flow Control : No
G - Software Flow Control : No

   Change which setting?
```

退出配置框只需连续按 ESC键即可返回。

软、硬盘及光驱的使用

在 Linux中对其他硬盘逻辑分区、软盘，光盘的使用与在 DOS与 Windows中的使用方法是不一样的，不能直接访问，因为在 Linux中它们都被视为文件，因此在访问使用前必须使用装载命令 mount将它们装载到系统的 /mnt目录中来，使用结束，必须进行卸载。命令格式如下：

mount – t文件系统类型 设备名 装载目录

文件类型常用的有：

msdos　　dos分区文件
ext2　　Linux的文件系统
swap　　Linux swap分区或 swap文件
iso9660　安装 CD-ROM的文件系统
vfat　　支持长文件名的 dos分区
hpfs　　OS/2分区文件系统

设备名是指要装载的设备的名称，如软盘、硬盘、光盘等，软盘

一般为 /dev/fd0 fd1，硬盘一般为 /dev/hda hdb，硬盘逻辑分区一般为期 hda1 hda2…等等，光盘一般为 /dev/hdc。在装载前一般要在 /dev/mnt目录下建立一个空的目录，如软盘为 floppy，硬盘分区为其盘符如 c，d等等，光盘为 cd-rom，使用命令：

mount–tmsdos /dev/fd0 /mnt/floppy

装载一个 mddos格式的软盘

mount–text2 /dev/fd0 /mnt/floppy

装载一个 Linux格式的软盘

mount–tvfat /dev/hda1 /mnt/c

装载 Windows98格式的硬盘分区

mount–t iso9660 /dev/hdc /mnt/cd-rom

装载一个光盘

Ubuntu 对于可移动设备能够实现 automount,mount 的地址在/media 下的一个子目录，具体地址可以输入 mount 命令查看。

装载完成之后便可对该目录进行操作，在使用新的软盘及光盘前必须退出该目录，使用卸载命令进行卸载，方可使用新的软盘及光盘，否则系统不会承认该软盘的，光盘在卸载前是不能用光驱面板前的弹出键退出的。

LILO与 GRUB

Linux安装时一般都安装 bootloader，可以支持多操作系统并存。常见的 bootloader有 LILO和 GRUB，Ubuntu默认使用的是 GRUB。

Diff创建软件补丁，用 patch打补丁

diff是生成源码补丁的必备工具。其命令格式为：

diff [命令行选项] 原始文件 新文件

常用命令行选项如下：

-r 递归处理目录 -u 输出统一格式 (unified format)

-N patch里包含新文件 -a patch里可以包含二进制文件

它的输出在 stdout上，所以你可能需要把它重定向到一个文件。

输出格式保存了上下文（缺省是上下各三行，最少需要两行），这样， patch的时候可以允许行号不精确匹配的情况出现。另外，在 patch文件的开头明确地用 --和 +++标示出原始文件和当前文件，也方便阅读。

通常，需要对整个软件包做修改，并生成一个 patch文件，下面是典型的操作过程。

tar xzvf software.tar.gz # 展开原始软件包，其目录为 software

cp _a software software-orig # 做个修改前的备份

cd software

[修改，测试……]

cd ..

diff -ruNa software-orig software > software-my.patch

现在就可以保存 software-my.patch做为这次修改的结果，

至于原始软件包，可以不必保存。等到下次需要再修改的时候，可以用 patch命令把这个补丁打进原始包，再继续工作。比如是在 Linux kernel上做的工作，就不必每次保存几十兆修改后的源码了。这是好处之一，好处之二是维护方便，由于 unified patch格式有一定的模糊匹配能力，能减少原软件包升级带来的维护工作量。

patch

patch程序根据补丁（patch）文件修改一个文件。补丁文件通常

是使用 diff程序建立的一个列表，这个列表包含如何修改原始文件的一些指令。由于节省时间和空间， Patch经常用于源代码的修补。可以想象一个有 1MB的程序包，这个程序包的下一个版本仅仅改变了前面一个版本的两个文件的内容，这个新版本可以完全以一个 1MB 的新版本进行发布或者以一个仅仅有 1KB 的补丁文件进行发布。这个补丁文件可以对第一个版本的进行更新，更新后的版本就和第二个版本完全一致了。因此，如果已经下载了第一个版本，那么为了下一个版本而进行的大数据量下载工作就可以有效地避免。

常用命令行选项：

patch [命令行选项] [待 patch的文件 [patch]]

-pn patch level(n是数字) -b[后缀] 生成备份，缺省是 .orig

为了说明什么是 patch level，这里看一个 patch文件的头标记。

diff -ruNa xc.orig/config/cf/Imake.cf xc.bsd/config/cf/Imake.cf

--- xc.orig/config/cf/Imake.cf Fri Jul 30 12:45:47 1999

+++ xc.new/config/cf/Imake.cf Fri Jan 21 13:48:44 2000

这个 patch如果直接应用，它会去找 xc.orig/config/cf目录下的 Imake.cf文件，假如你的源码树的根目录是缺省的 xc而不是 xc.orig，除了 mv xc xc.orig之外，有无简单的方法应用此 patch呢？ patch level就是为此而设： patch会把目标路径名砍去开头 patch level个节 (由 /分开的部分)。在本例中，可以用下述命令：

cd xc; patch -p1 < /pathname/xxx.patch

完成操作。注意，由于没有指定 patch文件， patch程序默认从 stdin读入，所以用了输入重定

向。

又例如：

diff -r dir1 dir2 >patch20020523.patch

递归的比较目录 dir1 与 dir2 内，所有各文件之不同处，并将不同处记录到 patch20020523.patch文件中。

patch -p1 < [patchfile]

-p1选项代表 patchfile中文件名左边目录的层数，顶层目录在不同的机器上有所不同。要使用这个选项，就要把你的 patch放在要被打补丁的目录下，然后在这个目录中运行 path -p1 < [patchfile]。

Linux下的硬盘分区

对习惯于使用 Dos或 Windows的用户来说，有几个分区就有几个驱动器，并且每个分区都会获得一个字母标识符，然后就可以选用这个字母来指定在这个分区上的文件和目录，它们的文件结构都是独立的，非常好理解。但对 Linux用户来说无论有几个分区，分给哪一目录使用，它归根结底就只有一个根目录，一个独立且唯一的文件结构。Linux 中每个分区都是用来组成整个文件系统的一部分，因为它采用了一种叫"载入"的处理方法，它的整个文件系统中包含了一整套的文件和目录，且将一个分区和一个目录联系起来。这时要载入的一个分区将使它的存储空间在一个目录下获得。

对于 IDE硬盘，驱动器标识符为"hdx~"其中"hd"表明分区所在设备的类型，这里是指 IDE硬盘了。"x"为盘号（a为基本盘，b为基本从属盘，c为辅助主盘，d为辅助从属盘），"~"代表分区，前四个分区用数字 1到 4表示，它们是主分区或扩展分区，从 5开始就是逻辑分区。例，hda3表示为第一个 IDE硬盘上的第三个主分区或扩展分区，hdb2表示为第二个 IDE硬盘上的第二个主分区或扩展分区。对于 SCSI硬盘则标识为"sdx~"，SCSI硬盘是用"sd"来表示分区所在设备的类型的，其余则和 IDE硬盘的表示方法一样。

从上面可以看到，Linux的分区是不同于其它操作系统分区的，它的分区格式只有 Ext2（3）和 Swap两种，Ext2（3）用于存放系统文件，Swap则作为 Linux的交换分区。Linux至少需要两个专门的分区（Linux Native和 Linux Swap）况且不能将 Linux安装在 Dos/Windows分区。一般来说将 Linux安装一个或多个类型为"Linux Native"的硬盘分区，但是在 Linux的每一个分区都必须要指定一个"MountPoint"（载入点），告诉 Linux在启动时，这个目录要给哪个目录使用。对"Swap"分区来说，一般定义一个且它不必要定义载入点。

SWAP分区是 Linux暂时存储数据的交换分区，它主要是把主内存上暂时不用的数据存起来，在需要的时候再调进内存内，且作为 SWAP使用的分区不用指定"Mout Point"（载入点），既然它作为交换分区，我们理所当然应给它指定大小，它至少要等于系统上实际内存的量，一般来说它的大小是内存的两倍，如果你是 16MB的内存，那么 SWAP分区的大小是 32MB左右，以此类推。但必须还要注意一点，SWAP分区不要大于 128MB，如果你是 64MB的内存，那么 SWAP分区最大也只能被定为 127MB，再大就是浪费空间了，因为系统不需要太大的交换分区。以此类推，如果你是 128MB或更大的内存，SWAP分区也只能最大被定为 127MB。可以创建和使用一个以上的交换分区，最多 16个。

Linux Native是存放系统文件的地方，它只能用 EXT2（3）的分区类型。对 Windows用户来说，操作系统必须装在同一分区里，它是商业软件吗？所以你没有选择的余地！对 Linux 来说，你有了较大的选择余地，你可以把系统文件分几个区来装（必须要说明载入点），也可以就装在同一个分区中（载入点是"/"）。

/boot分区，它包含了操作系统的内核和在启动系统过程中所要用到的文件，建这个分区是有必要的，因为目前大多数的 PC机要受到 BIOS的限制，况且如果有了一个单独的 /boot启动分区，即使

主要的根分区出现了问题，计算机依然能够启动。这个分区的大小约在 50MB—100MB之间。但是如果想用 LILO启动 Linux系统的话，含有 /boot的分区必须完全在柱面 1023以下。又由于 8GB后的数据 LILO不能读取，所以 Linux要安装在 8GB的区域以内。

/usr分区，是 Linux系统存放软件的地方，如有可能应将最大空间分给它。

/home分区，是用户的 home目录所在地，这个分区的大小取决于有多少用户。如果是多用户共同使用一台电脑的话，这个分区是完全有必要的，况且根用户也可以很好地控制普通用户使用计算机，如对用户或者用户组实行硬盘限量使用，限制普通用户访问哪些文件等。其实单用户也有建立这个分区的必要，因为没这个分区的话，那么你只能以 root用户的身份登陆系统，这样做是危险的，因为 root用户对系统有绝对的使用权，一旦对系统进行了误操作，就会导致系统崩溃。

/var/log 分区，是系统日志记录分区，如果设立了这一单独的分区，这样即使系统的日志文件出现了问题，它们也不会影响到操作系统的主分区。

/tmp分区，用来存放临时文件。这对于多用户系统或者网络服务器来说是有必要的。这样即使程序运行时生成大量的临时文件，或者用户对系统进行了错误的操作，文件系统的其它部分仍然是安全的。因为文件系统的这一部分仍然还承受着读写操作，所以它通常会比其它的部分更快地发生问题。

/bin分区，存放标准系统实用程序。

/dev分区，存放设备文件。

/opt分区，存放可选的安装的软件。

/sbin分区，存放标准系统管理可执行文件，如 insmod, ifconfig等。

上面介绍了几个常用的分区，一般来说需要一个 SWAP分区，一个 /boot分区，一个 /usr分区，一个 /home 分区，一个 /var/log 分区。当然这没有什么规定，完全是依照个人来定的。但记住至少要有两个分区，一个 SWAP分区，一个 分区。

用户可以使用 fdisk分区工具：

Fdisk：它是传统的 Linux硬盘分区工具，比 DiskDruid更强大，使用更加灵活。但是 Fdisk要求用户对硬盘分区有一定经验，并能够适应且读懂简单的文本界面。如果你是第一次对一个硬盘驱动器进行分区操作的话，最好还是避免 Fdisk这样的程序，它虽然强大但用起来的感觉不是太好的。

附录 C gcc 与 gdb

gcc 是 GNU 的 C 和 C++ 编译器，它是 Linux 中最重要的软件开发工具。实际上，gcc 能够编译三种语言：C、C++ 和 ObjectC（C 语言的一种面向对象扩展）。利用 gcc 命令可同时编译并连接 C 和 C++ 源程序。 汇编语言的编译器为 as

编译器被成功的移植到不同的处理器平台上。标准 PC Linux上的 gcc是 FOR INTEL CPU的，而华恒 HHPC852系列开发套件使用的是 FOR powerpc系列处理器的 gcc编译器 powerpc-Linux-gcc 和 powerpc-Linux-as及其相应的 GNU Binutils工具集（如 ld链接工具，objcopy、objdump等工具）。

gcc 命令的常用选项有：

选项	说明
-ansi	只支持 ANSI 标准的 C 语法。这一选项将禁止 GNU C 的某些特色，例如 asm 或 typeof 关键词。
-c	只编译并生成目标文件。
-DMACRO	以字符串 “1” 定义 MACRO 宏。
-DMACRO=DEFN	以字符串 “DEFN” 定义 MACRO 宏。
-E	只运行 C 预编译器。
-g	生成调试信息。GNU 调试器可利用该信息。
-IDIRECTORY	指定额外的头文件搜索路径 DIRECTORY。
-LDIRECTORY	指定额外的函数库搜索路径 DIRECTORY。
-lLIBRARY	连接时搜索指定的函数库 LIBRARY。
-m486	针对 486 进行代码优化。
-o FILE	生成指定的输出文件。用在生成可执行文件时。
-O0	不进行优化处理。
-O 或 -O1	优化生成代码。
-O2	进一步优化。
-O3	比 -O2 更进一步优化，包括 inline 函数。
-shared	生成共享目标文件。通常用在建立共享库时。
-static	禁止使用共享连接。
-UMACRO	取消对 MACRO 宏的定义。
-w	不生成任何警告信息。
-Wall	生成所有警告信息。

ld文件

编译完成之后，就要执行 ld进行链接。 ld工具处理 ld文件。

ld文件采用 AT&T链接命令语言写成 用于控制整个链接过程。

GDB

Linux 包含了一个叫 gdb 的 GNU 调试程序 . gdb 是一个用来调试 C 和 C++ 程序的强力调试器 . 它使你能在程序运行时观察程序的内部结构和内存的使用情况 .Gdb功能非常强大：

可监视程序中变量的值 .

可设置断点以使程序在指定的代码行上停止执行 .

支持单步执行等

在命令行上键入 gdb 并按回车键就可以运行 gdb 了，如果一切正常的话，gdb 将被启动并且你将在屏幕上看到类似的内容：

```
GNU gdb 6.8-debian
Copyright (C) 2008 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i486-linux-gnu".
.
(gdb)
```

当你启动 gdb 后，你能在命令行上指定很多的选项.你也可以以下面的方式来运行 gdb：

gdb <fname>

当你用这种方式运行 gdb，你能直接指定想要调试的程序.这将告诉 gdb 装入名为 fname 的可执行文件.你也可以用 gdb 去检查一个因程序异常终止而产生的 core 文件，或者与一个正在运行的程序相连.你可以参考 gdb 指南页或在命令行上键入 gdb -h 得到一个有关这些选项的说明的简单列表.

为了使 gdb 正常工作，你必须使你的程序在编译时包含调试信息.调试信息包含你程序里的每个变量的类型和在可执行文件里的地址映射以及源代码的行号. gdb 利用这些信息使源代码和机器码相关联.

在编译时用 -g 选项打开调试选项.

gdb 支持很多的命令使你能实现不同的功能.这些命令从简单的文件装入到允许你检查所调用的堆栈内容的复杂命令，下面列出了你在用 gdb 调试时会用到的一些命令.想了解 gdb 的详细使用请参考 gdb 的指南页.

gdb 的常用命令

命令	说明
break NUM	在指定的行上设置断点。
bt	显示所有的调用栈帧。该命令可用来显示函数的调用顺序。
clear	删除设置在特定源文件、特定行上的断点。其用法为：clear FILENAME:NUM。
continue	继续执行正在调试的程序。该命令用在程序由于处理信号或断点而导致停止运行时。
display EXPR	每次程序停止后显示表达式的值。表达式由程序定义的变量组成。
file FILE	装载指定的可执行文件进行调试。
help NAME	显示指定命令的帮助信息。
info break	显示当前断点清单，包括到达断点处的次数等。
info files	显示被调试文件的详细信息。
info func	显示所有的函数名称。
info local	显示当函数中的局部变量信息。
info prog	显示被调试程序的执行状态。
info var	显示所有的全局和静态变量名称。
kill	终止正被调试的程序。
list	显示源代码段。

make 在不退出 gdb 的情况下运行 make 工具。
next 在不单步执行进入其他函数的情况下，向前执行一行源代码。
print EXPR 显示表达式 EXPR 的值。

gdb 支持很多与 UNIX shell 程序一样的命令编辑特征 .你能象在 bash 或 tcsh 里那样按 Tab 键让 gdb 帮你补齐一个唯一的命令 ,如果不唯一的话 gdb 会列出所有匹配的命令 .你也能用光标键上下翻动历史命令 .

gdb 应用举例

下面用一个实例教你一步步的用 gdb 调试程序 .被调试的程序相当的简单 ,但它展示了 gdb 的典型应用 .

下面列出了将被调试的程序 .这个程序被称为 greeting ,它显示一个简单的问候 ,再用反序将它列出 .

```
#include <stdio.h>
main (){
   char my_string[] = "hello there";
   my_print (my_string);
   my_print2 (my_string);
}
void my_print (char *string){
   printf ("The string is %s\n", string);
}
void my_print2 (char *string){
   char *string2;
   int size, i;
   size = strlen (string);
   string2 = (char *)malloc (size + 1);
   for (i = 0; i < size; i++)
      string2[size - i] = string[i];
   string2[size+1] = `\0';
   printf ("The string printed backward is %s\n", string2);
}
```

用下面的命令编译它 :

gcc -o test test.c

这个程序执行时显示如下结果 :

The string is hello there
The string printed backward is

输出的第一行是正确的 ,但第二行打印出的东西并不是我们所期望的 .我们所设想的输出应该是 :

The string printed backward is erehtolleh

由于某些原因 ,my_print2 函数没有正常工作 .让我们用 gdb 看看问题究竟出在哪儿 ,先键入如下命令 :

gdb greeting

【注意】!

记得在编译 greeting 程序时把调试选项打开 .

如果你在输入命令时忘了把要调试的程序作为参数传给 gdb ，你可以在 gdb 提示符下用 file 命令来载入它：

(gdb)**file greetin** g

这个命令将载入 greeting 可执行文件就像你在 gdb 命令行里装入它一样 . 这时你能用 gdb 的 run 命令来运行 greeting 了 . 当它在 gdb 里被运行后结果大约会象这样：

(gdb)**run**

Starting program: /root/greeting

The string is hello there

The string printed backward is

Program exited with code 041

这个输出和在 gdb 外面运行的结果一样 . 问题是，为什么反序打印没有工作？为了找出症结所在，我们可以在 my_print2 函数的 for 语句后设一个断点，具体的做法是在 gdb 提示符下键入 list 命令三次，列出源代码：

(gdb)**list**

(gdb) 回车

(gdb) 回车

（在 gdb 提示符下按回车健将重复上一个命令。）

要 list三次是因为一次无法显示全部文件内容，而必须多次才能翻到想要设置断点的文件行处。根据列出的源程序，能看到要设断点的地方在第 24 行，在 gdb 命令行提示符下键入如下命令设置断点：

(gdb)**break 24**

gdb 将作出如下的响应：

Breakpoint 1 at 0x139: file greeting.c, line 24

(gdb)

现在再键入 run 命令，将产生如下的输出：

Starting program: /root/greeting

The string is hello there

Breakpoint 1, my_print2 (string = 0xbfffdc4 "hello there") at greeting.c :24

24 string2[size-i]=string[i]

通过设置一个观察 string2[size - i] 变量的值的观察点来看出错误是怎样产生的，做法是键入：

(gdb)**watch string2[size - i]**

gdb 将作出如下回应：

Watchpoint 2: string2[size - i]

现在可以用 next 命令来一步步的执行 for 循环了：

(gdb)**next**

经过第一次循环后， gdb 告诉我们 string2[size - i] 的值是 `h`. gdb 用如下的显示来告诉你这个信息：

Watchpoint 2, string2[size - i]

Old value = 0 `\000'

New value = 104 `h'

my_print2(string = 0xbfffdc4 "hello there") at greeting.c:23

23 for (i=0; i<size; i++)

这个值正是期望的.后来的数次循环的结果都是正确的.当 i=10 时,表达式 string2[size - i] 的值等于 `e`, size - i 的值等于 1,最后一个字符已经拷到新串里了.

如果你再把循环执行下去,你会看到已经没有值分配给 string2[0] 了, 而它是新串的第一个字符,因为 malloc 函数在分配内存时把它们初始化为空(null)字符.所以 string2 的第一个字符是空字符.这解释了为什么在打印 string2 时没有任何输出了.

现在找出了问题出在哪里,修正这个错误是很容易的.你得把代码里写入 string2 的第一个字符的的偏移量改为 size - 1 而不是 size.这是因为 string2 的大小为 12,但起始偏移量是 0,串内的字符从偏移量 0 到 偏移量 10,偏移量 11 为空字符保留.

附录 D Makefile

在大型的开发项目中，通常有几十到上百个的源文件，如果每次均手工键入 gcc 命令进行编译的话，则会非常不方便。因此，人们通常利用 make 工具来自动完成编译工作。这些工作包括：如果仅修改了某几个源文件，则只重新编译这几个源文件；如果某个头文件被修改了，则重新编译所有包含该头文件的源文件。利用这种自动编译可大大简化开发工作，避免不必要的重新编译。

实际上，make 工具通过一个称为 makefile 的文件来完成并自动维护编译工作。makefile 需要按照某种语法进行编写，其中说明了如何编译各个源文件并连接生成可执行文件，并定义了源文件之间的依赖关系。

当修改了其中某个源文件时，如果其他源文件依赖于该文件，则也要重新编译所有依赖该文件的源文件。

makefile 文件是许多编译器，包括 WindowsNT 下的编译器维护编译信息的常用方法，只是在集成开发环境中，用户通过友好的界面修改 makefile 文件而已。默认情况下，GNU make 工具在当前工作目录中按如下顺序搜索 makefile：

*GNUmakefile

*makefile

*Makefile

在 UNIX 系统中，习惯使用 Makefile 作为 makfile 文件。如果要使用其他文件作为 makefile，则可利用类似下面的 make 命令选项指定 makefile 文件：

$make -fMakefile.debug

Makefile 基本结构

makefile 中一般包含如下内容：

* 需要由 make 工具创建的项目，通常是目标文件和可执行文件。通常使用“目标（target）”一词来表示要创建的项目。

* 要创建的项目依赖于哪些文件。

* 创建每个项目时需要运行的命令。

例如，假设你现在有一个 C++ 源文件 test.C，该源文件包含有自定义的头文件 test.h，则目标文件 test.o 明确依赖于两个源文件：test.c 和 test.h。另外，你可能只希望利用 g++ 命令来生成 test.o 目标文件。

这时，就可以利用如下的 makefile 来定义 test.o 的创建规则：

```
#This makefile just is a example.
#The following lines indicate how test.o depends
# test.C and test.h, and how to create test.o

test.o: test.c test.h
    g++ -c -g test.C
```

从上面的例子注意到，第一个字符为 # 的行为注释行。第一个非注释行指定 test.o 为目标，并且依赖于 test.c 和 test.h 文件。随后的行指定了如何从目标所依赖的文件建立目标。当 test.c 或 test.h 文件在编译之后又被修改，则 make 工具可自动重新编译 test.o，如果在前后两次编译之间，test.C 和 test.h 均没有被修改，而且 test.o 还存在的话，就没有必要重新编译。这种依赖关系在多源文件的程序编译中尤其重要。通过这种依赖关系的定义，make 工具可避免许多不必要的编译工作。当然，利用 Shell 脚本也可以达到自动编译的效果，但是，Shell 脚本将全部编译任何源文件，

包括哪些不必要重新编译的源文件，而 make 工具则可根据目标上一次编译的时间和目标所依赖的源文件的更新时间而自动判断应当编译哪个源文件。

一个 makefile 文件中可定义多个目标，利用 make target 命令可指定要编译的目标，如果不指定目标，

则使用第一个目标。通常，makefile 中定义有 clean 目标，可用来清除编译过程中的中间文件，例如：

```
clean:
    rm -f *.o
```

运行 make clean 时，将执行 rm -f *.o 命令，最终删除所有编译过程中产生的所有中间文件。

Makefile 变量

GNU 的 make 工具除提供有建立目标的基本功能之外，还有许多便于表达依赖性关系以及建立目标的命令的特色。其中之一就是变量或宏的定义能力。如果你要以相同的编译选项同时编译十几个 C 源文件，而为每个目标的编译指定冗长的编译选项的话，将是非常乏味的。但利用简单的变量定义，可避免这种乏味的工作：

```
#Define macros for name of compiler
CC = gcc

#Define a macro for the CC flags
CCFLAGS = -D_DEBUG -g -m486
#A rule for building a object file
test.o: test.c test.h
    $(CC) -c $(CCFLAGS) test.c
```

在上面的例子中，CC 和 CCFLAGS 就是 make 的变量。GNU make 通常称之为变量，而其他 UNIX 的 make 工具称之为宏，实际是同一个东西。在 makefile 中引用变量的值时，只需变量名之前添加 $ 符号，如上面的 $(CC) 和 $(CCFLAGS)。

GNU make 的主要预定义变量 i

GNU make 有许多预定义的变量，这些变量具有特殊的含义，可在规则中使用。表 1-5 给出了一些主要的

预定义变量，除这些变量外，GNU make 还将所有的环境变量作为自己的预定义变量。

$* 不包含扩展名的目标文件名称。

$+ 所有的依赖文件，以空格分开，并以出现的先后为序，可能包含重复的依赖文件。

$< 第一个依赖文件的名称。

$? 所有的依赖文件，以空格分开，这些依赖文件的修改日期比目标的创建日期晚。

$@ 目标的完整名称。

$^ 所有的依赖文件，以空格分开，不包含重复的依赖文件。

$% 如果目标是归档成员，则该变量表示目标的归档成员名称。例如，如果目标名称为 mytarget.so(image.o)，则 $@ 为 mytarget.so，而 $% 为 image.o。

AR 归档维护程序的名称，默认值为 ar。

ARFLAGS 归档维护程序的选项。

AS	汇编程序的名称，默认值为 as。
ASFLAGS	汇编程序的选项。
CC	C 编译器的名称，默认值为 cc。
CCFLAGS	C 编译器的选项。
CPP	C 预编译器的名称，默认值为 $(CC) -E。
CPPFLAGS	C 预编译的选项。
CXX	C++ 编译器的名称，默认值为 g++。
CXXFLAGS	C++ 编译器的选项。
FC	FORTRAN 编译器的名称，默认值为 f77。
FFLAGS	FORTRAN 编译器的选项。

隐含规则

GNU make 包含有一些内置的或隐含的规则，这些规则定义了如何从不同的依赖文件建立特定类型的目标。

GNU make 支持两种类型的隐含规则：

* 后缀规则（Suffix Rule）。后缀规则是定义隐含规则的老风格方法。后缀规则定义了将一个具有某个后缀的文件（例如，.c 文件）转换为具有另外一种后缀的文件（例如，.o 文件）的方法。每个后缀规则以两个成对出现的后缀名定义，例如，将 .c 文件转换为 .o 文件的后缀规则可定义为：

```
.c.o:
$(CC) $(CCFLAGS) $(CPPFLAGS) -c -o $@ $<
```

* 模式规则（pattern rules）。这种规则更加通用，因为可以利用模式规则定义更加复杂的依赖性规则。模式规则看起来非常类似于正则规则，但在目标名称的前面多了一个 % 号，同时可用来定义目标和依赖文件之间的关系，例如下面的模式规则定义了如何将任意一个 X.c 文件转换为 X.o 文件：

```
%.c:%.o
$(CC) $(CCFLAGS) $(CPPFLAGS) -c -o $@ $<
```

Makefile 范例

运行 make

我们知道，直接在 make 命令的后面键入目标名可建立指定的目标，如果直接运行 make，则建立第一个目标。我们还知道可以用 make -f mymakefile 这样的命令指定 make 使用特定的 makefile，而不是默认的 GNUmakefile、makefile 或 Makefile。但 GNU make 命令还有一些其他选项，下面给出了这些选项。

-C DIR	在读取 makefile 之前改变到指定的目录 DIR。
-f FILE	以指定的 FILE 文件作为 makefile。
-h	显示所有的 make 选项。
-i	忽略所有的命令执行错误。
-I DIR	当包含其他 makefile 文件时，可利用该选项指定搜索目录。
-n	只打印要执行的命令，但不执行这些命令。
-p	显示 make 变量数据库和隐含规则。
-s	在执行命令时不显示命令。
-w	在处理 makefile 之前和之后，显示工作目录。
-W FILE	假定文件 FILE 已经被修改。

附录 E Qtopia 移植

华恒开发平使用 Qt作为 GUI，其代码目录为 applications/ qtopia-free-2.2.0.本移植文档针对 qtopia-free-2.2.0，但移植方法同样适用于 qte4，qtopia4 core，qtopia-free-2.2.0 等版本。6410-integration-R2上 qtopia-free-2.2.0。用户可参考此处说明。

（一）编译相关库

1.e2fs

./configure --host=arm-none-linux-gnueabi --enable-elf-shlibs --with-cc=arm-none-linux-gnueabi-gcc --with-linker=arm-none-linux-gnueabi-ld --prefix=/opt/arm-2008q3/arm-none-linux-gnueabi

make

make install

注意：--prefix=/opt/arm-2008q3/arm-none-linux-gnueabi 是将生成的库安装到编译器的库目录下

2.jpeg

./configure –enable-shared

vi Makefile 将 CC AR AR2 改成一下：

CC= arm-none-linux-gnueabi-gcc
AR= arm-none-linux-gnueabi-ar rc
AR2=arm-none-linux-gnueabi-ranlib
prefix=/opt/arm-2008q3/arm-none-linux-gnueabi

make

make install

3.zlib

./configure -shared

vi Makefile 将 CC AR LDSHARED CPP RANLIB prefix 改成以下：

CC=/opt/arm-2008q3/bin/arm-none-linux-gnueabi-gcc
LDSHARED=/opt/arm-2008q3/bin/arm-none-linux-gnueabi-gcc -shared -Wl,-soname,libz.so.1
CPP=/opt/arm-2008q3/bin/arm-none-linux-gnueabi -E
AR=/opt/arm-2008q3/bin/arm-none-linux-gnueabi-ar rc
RANLIB=/opt/arm-2008q3/bin/arm-none-linux-gnueabi-ranlib
prefix =/opt/arm-2008q3/arm-none-linux-gnueabi

make

make install

4.libpng

./configure --host=arm-none-linux-gnueabi --prefix=/opt/arm-2008q3/bin/arm-none-linux-gnueabi

viMakefile将 CC AR RC RANLIB prefix 改成以下

CC=arm-none-linux-gnueabi-gcc
AR_RC=arm-none-linux-gnueabi-ar rc
RANLIB=arm-none-linux-gnueabi-ranlib
prefix =/opt/arm-2008q3/arm-none-linux-gnueabi

make

make install

5.tslib（触摸屏校准）

./autogen.sh

echo "ac_cv_func_malloc_0_nonnull=yes" > arm-linux.cache

./configure --host=arm-none-linux-gnueabi --target=arm-none-linux-gnueabi --enable-inputapi=no --prefix=/opt/arm-2008q3/arm-none-linux-gnueabi

注意：有时会出现错误提示：undefined reference to rpl_malloc

解决：vi config.h.in 中的# undef malloc 注释掉即可

为了防止出现"libtool:link: only absolute run-paths are allowed"的错误，在/tslib/plugins/Makefile 里面找 rpath，找到将其注释并加上绝对路径

#LDFLAGS :=$(LDFLAGS) -rpath $(PLUGIN_DIR)

LDFLAGS :=$(LDFLAGS) -rpath `cd $(PLUGIN_DIR) && pwd`

make

make install

（二）编译 qtopia-free-2.2.0

exportMYHOME=/home/ccs/HHARM6410/applications/ （路经根据自己实际修改）

修改文件

vi $MYHOME/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++/qmake.conf

将此行

QMAKE_LIBS_QT = -lqte

修改为

QMAKE_LIBS_QT= -lqte -lpng -lts -lz -luuid –ljpeg 这里增加了另外的库

如果想让 Qtopia自己支持触摸屏，需要修改 $QPEDIR/qtopia/src/qt/qconfig-qpe.h文件，定义相应的宏，在最后加上

#define QT_QWS_IPAQ

#define QT_QWS_IPAQ_RAW

在最后运行的时候，加上如下环境变量：

export QWS_MOUSE_PROTO=TPanel:/dev/h3600_tsraw（触摸屏具体名称和位置可能需要根据实际情况修改）

如果想让 Qtopia支持鼠标，需要修改 $QPEDIR/qtopia/src/qt/qconfig-qpe.h文件

注释如下部分：

```
/*
#ifndef QT_NO_QWS_CURSOR
#define QT_NO_QWS_CURSOR
#endif
#ifndef QT_NO_QWS_MOUSE_AUTO
#define QT_NO_QWS_MOUSE_AUTO
#endif
#ifndef QT_NO_QWS_MOUSE_PC
#define QT_NO_QWS_MOUSE_PC
#endif
*/
```

在最后运行的时候，加上如下环境变量：

export QWS_MOUSE_PROTO=TPanel:/dev/h3600_tsraw（触摸屏具体名称和位置可能需要根据实际情况修改）

export QWS_MOUSE_PROTO=USB:/dev/input/mice

如果想让 Qtopia支持 USB标准键盘，在最后运行的时候，加上如下环境变量：

export QWS_KEYBOARD=USB:/dev/input/event0

（2）设置环境变量：

export QPEDIR=$MYHOME/qtopia-free-2.2.0/qtopia

export QTDIR=$MYHOME/qtopia-free-2.2.0/qt2

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$QTDIR/lib:$QPEDIR/lib:/home/qtopia-arm-home/arm/lib:/opt/arm-2008q3/arm-none-linux-gnueabi/lib

export TMAKEDIR=$MYHOME/qtopia-free-2.2.0/tmake

export TMAKEPATH=$TMAKEDIR/lib/qws/linux-arm-g++

（3）准备配置文件：

cp $QPEDIR/src/qt/qconfig-qpe.h $QTDIR/src/tools

cd $QPEDIR/src/libraries/qtopia

cp custom-linux-ipaq-g++.cpp custom-linux-arm-g++.cpp

cp custom-linux-ipaq-g++.h custom-linux-arm-g++.h

（4）开始 configure

cd -

./configure -qte "-embedded -no-xft -qconfig qpe -no-qvfb -depths 16,24 -system-jpeg -system-libpng -system-zlib -tslib -gif -thread -no-xft -lz -lpng -lts -luuid -ljpeg -release -platform linux-g++ -xplatform linux-arm-g++" -qpe '-edition pda -no-qtopiadesktop -prefix=/mnt/nand/qtopia-2.2.0 -release -platform linux-g++ -xplatform linux-arm-g++' -qt2 '-xplatform linux-arm-g++ -platform linux-g++' -dqt '-thread -qt-gif -fast -platform linux-g++'

『说明』

（1）华恒在编译上述的库时已经将生产的库文件安装到编译器的库中了，生成的 QT 文件在 **/mnt/nand/qtopia-2.2.0 下，由上述可知。**

（2）qtopia 的配置文件$QPEDIR/src/libraries/qtopia/src/qt/qconfig-qpe.h 说明：

配置文件 qconfig-qpe.h中，注释掉 QT_NO_QWS_CURSOR那行，其他可以根据需要注释某些宏，另外在这个文件中添加 #define QT_QWS_IPAQ使触摸屏使用相应定义的数据结构 .

cp $QPEDIR/src/qt/qconfig-qpe.h $QTDIR/src/tools

这样在 configure 时就可以通过 -qconfig qpe 指定使用这个配置文件了。另外还有个文件 $MYHOME/qtopia-free-2.2.0/qtopia/mkspecs/qws/linux-arm-g++/qmake.conf

也是控制库裁减的，可自行修改。注释掉

define QT_NO_QWS_CURSOR一行。

另外如果加了 libuuid和 libjpeg等其他库的话，则把

QMAKE_LIBS_QT = -lqte改为 QMAKE_LIBS_QT = -lqte -lpng -lts -lz -luuid –ljpeg

开始配置，编译：

./configure -qte "-embedded -no-xft -qconfig qpe -no-qvfb -depths 16,24 -system-jpeg -system-libpng -system-zlib -tslib -gif -thread -no-xft -lz -lpng -lts -luuid -ljpeg -release -platform linux-g++ -xplatform linux-arm-g++" -qpe '-edition pda -no-qtopiadesktop -prefix=/mnt/nand/qtopia-2.2.0 -release -platform linux-g++ -xplatform linux-arm-g++' -qt2 '-xplatform linux-arm-g++ -platform linux-g++' -dqt '-thread -qt

-gif -fast -platform linux-g++'

上面是配置选项，可以 -help查看说明，简单说一下配置：
-prefix=/mnt/nand/qtopia-2.2.0指定 make install时安装目录
-noqvfb因为使用 framebuffer，所以不需要 qvfb
-extra-qtopiacore-config指定 qtopia-core的编译配置：
-xplatform选定编译配置
-qconfig选定 qconfig-xxx.h
-depths指定屏的深度
然后 make;make install。
（3）触摸屏校准：
触摸屏校准可以在开发板上执行：
/hhtech/app/ts_calibrate
用触摸笔按屏上提示的点操作后会生成校准文件 /etc/pointical，华恒提供的包中已经包括针对屏的校准文件，即 romfs/etc/pointical，一般不需要重新校准。如果有需要请删除 pointical再自行校准。
(4)运行 Qtopia：
把生成的 /mnt/nand/qtopi-free-2.2.0文件夹放到 sd卡或 nand中，并 mount到 /mnt/nand目录（其他目录也可以，不过需要 ln -s创建链接，视你的 -prefix参数而定），在开发板上：
env QTDIR=/mnt/qtopia-free-2.2.0 LD_LIBRARY_PATH=/mnt/qtopia-free-2.2.0/lib QWS_MOUSE_PROTO=linuxtp:/dev/ts0 QWS_DISPLAY="linuxfb:mmWidth35:mmHeight45:0" /mnt/qtopia-free-2.2.0/bin/qpe –qws
或 /hhtech/app/runqpe（上述命令编成的脚本）
即可在屏上显示 qpe程序。qtopia已经烧到 nand flash中，可在开发板启动后运行脚本启动 qpe；qtopia需要在内核中打开 system v ipc支持；
qtopia运行后会在环境变量 HOME指定的目录下创建一些文件，如果运行出现错误，请检查此目录内容和权限。

附录 F 利用仿真器(Banyan+axd)烧写引导代码

此方法大致有以下几个步骤：
注：目前国内市场支持 arm1176的仿真器不多华恒科技使用 Banyan-u一款国产仿真器。
1.选择从 nand falsh启动 (出厂默认设置)参考 5.7 外设接口图；
2.连接开发板和 PC机串口 115200（请确认已经安装好 Banyan仿真器驱动以及 ADS1.2包含 AXD工具）
3.连接仿真器后打开电源，打开 DaemonU.exe，仿真器会自动查找 CPU如下图 a所示，显示连接成功后，打开 AXD调试器，按图 b所示方法使 AXD连接 Banyan仿真器，使用快捷键 alt+1打开 Command Line窗口，ctrl+I打开 memory窗口，可以在地址栏敲入 0x8000000,0x50001000等地址查询相应地址内容（第一次使用才需要这一步）清除 cpu状态如图 c所示。
4.按第 3步设置好后关闭 AXD 再并重新打开 AXD 迅速在 Command Line窗口敲入
obey 6410_init.txt（请将 6410_init.txt文件拷贝到 ADS安装目录的 BIN文件夹下如本机安装位置 D:\Program Files\ARM\ADSv1_2\Bin，6410_init.txt 内容有一行 lb "E:\work\S3C6410\u-boot\RAM_UBOOT.bin", 0x50001000，请确保 RAM_UBOOT.bin在所指定目录或者请自行修改）

5运行第 4步需要 1分钟左右时间，请在串口下等待运行完第 4步需要迅速在串口下按回车等任意键，让其停在 u-boot的提示符下，运行完成后即可在串口窗口看到 u-boot已经运行起来，此 u-boot-ram.bin是 RAM 版的 u-boot，只能通过仿真器下载到 RAM 中运行的，并不能真正的烧写到 flash中去，运行以下命令，把真正烧写到 flash中去的 ROM 版 u-boot.bin烧写到 flash中去。

tftp 53000000 u-boot.bin

nand erase 0 40000

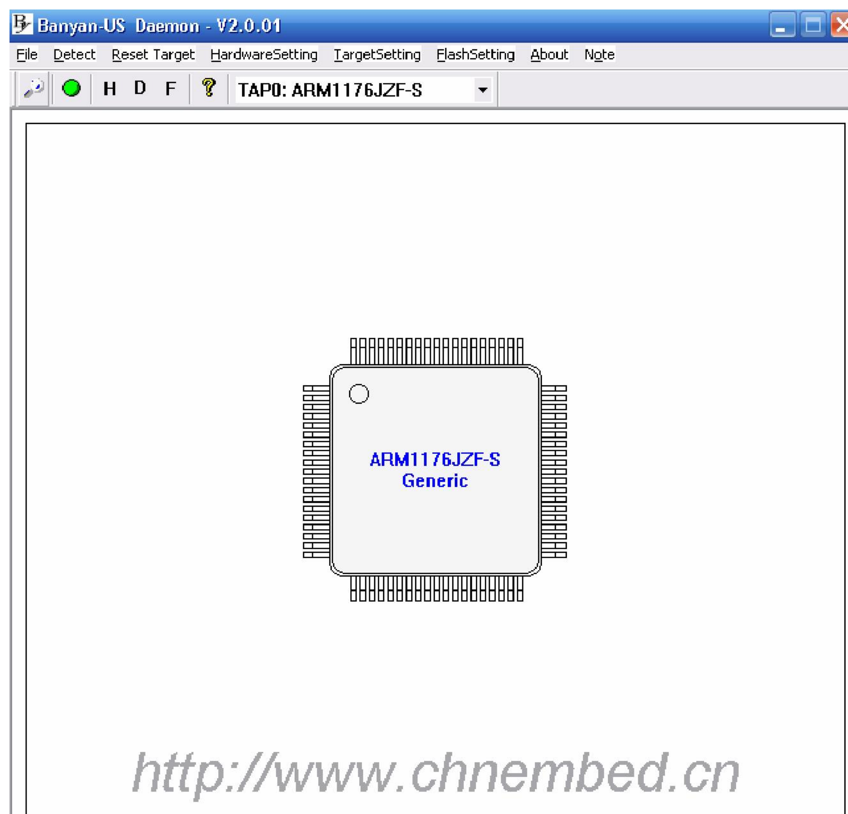
nand write 53000000 0 40000

6完成烧写后关闭电源，拆下仿真器。启动开发板已经运行更新的程序。

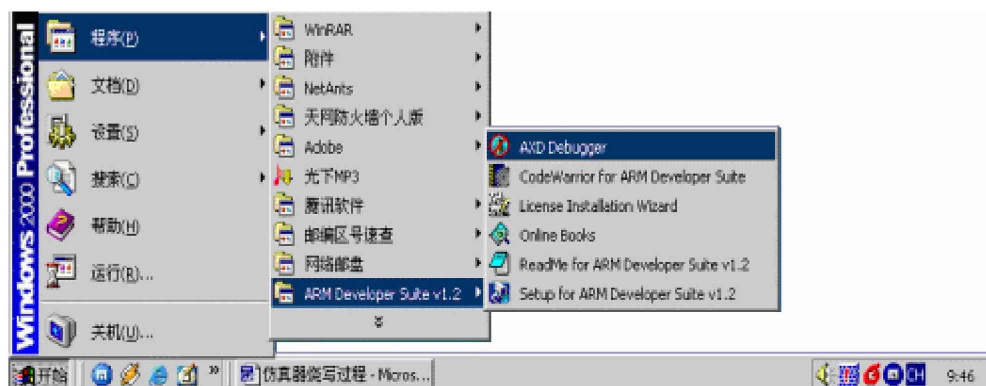
【注意】！第 4 步敲入命令请尽量快，可以从其他文档直接拷贝 obey 6410_init.txt 命令到命令行迅速按回车运行

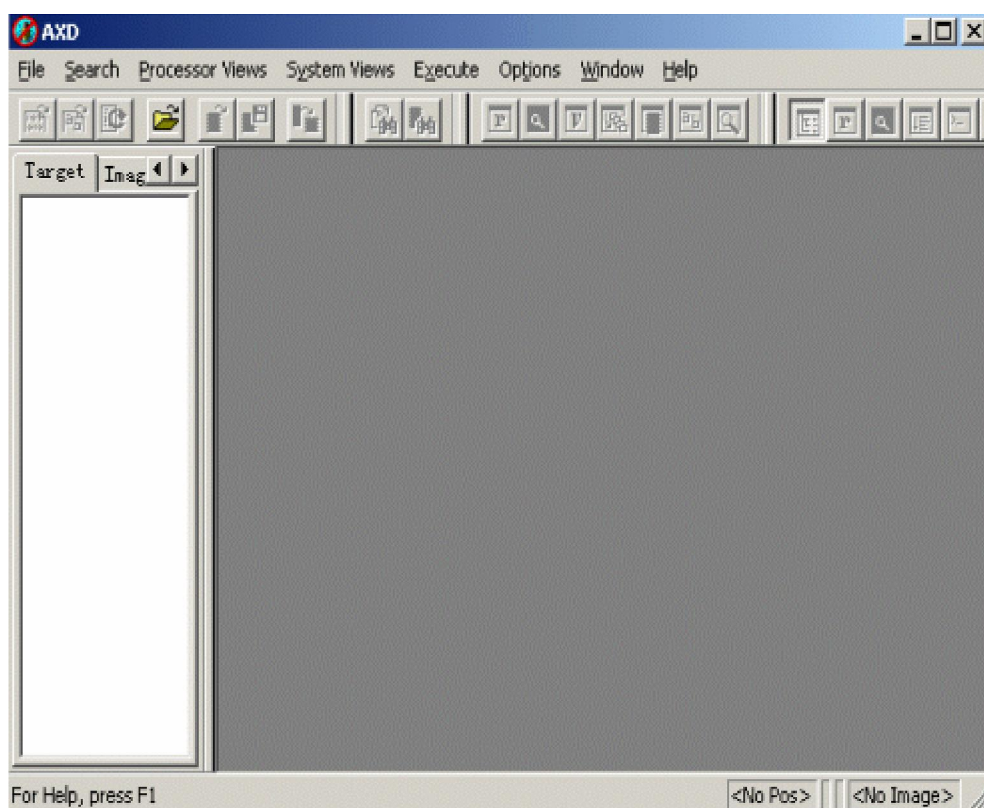
下面是仿真器连接时的一些界面，一般情况下，默认配置为“ Auto-Configure”打开后就能进行与仿真器之间的连接。

a.仿真器与 cpu之间连接成功

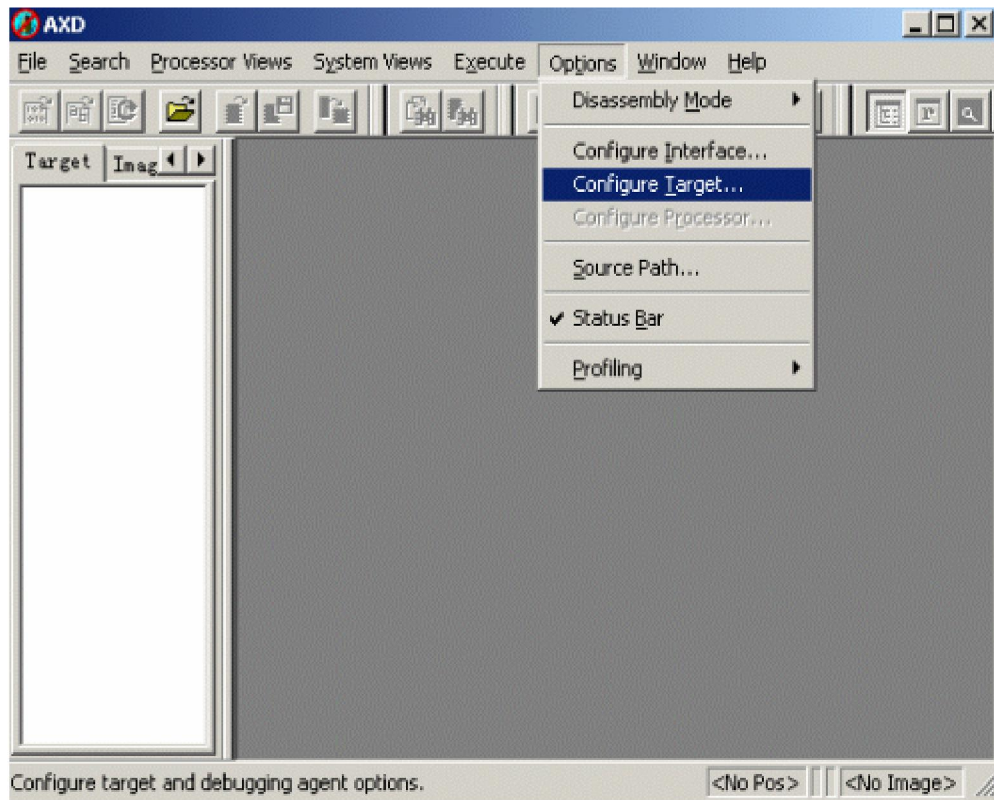


b. 打开 ARM Developer Suite，建立与仿真器的连接。

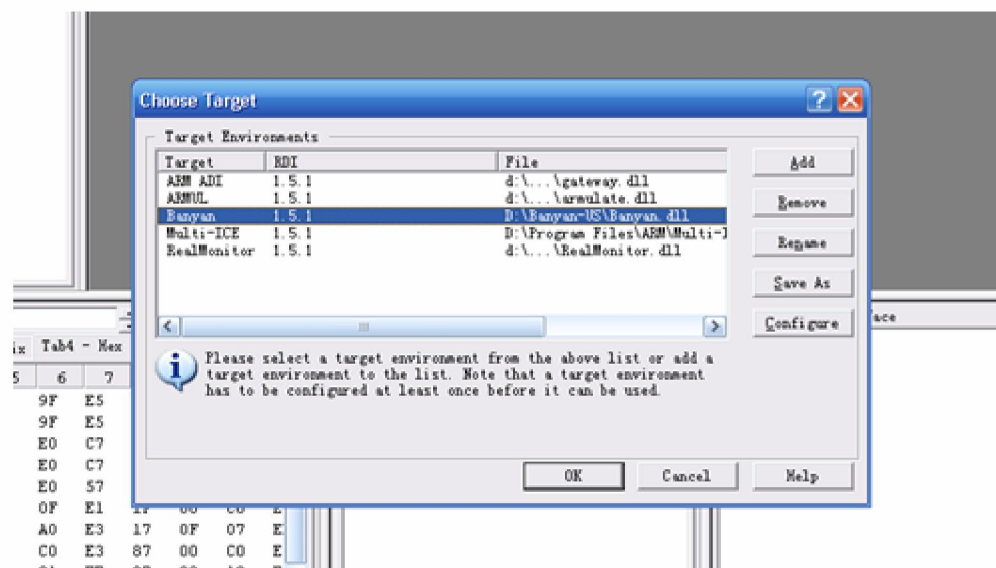




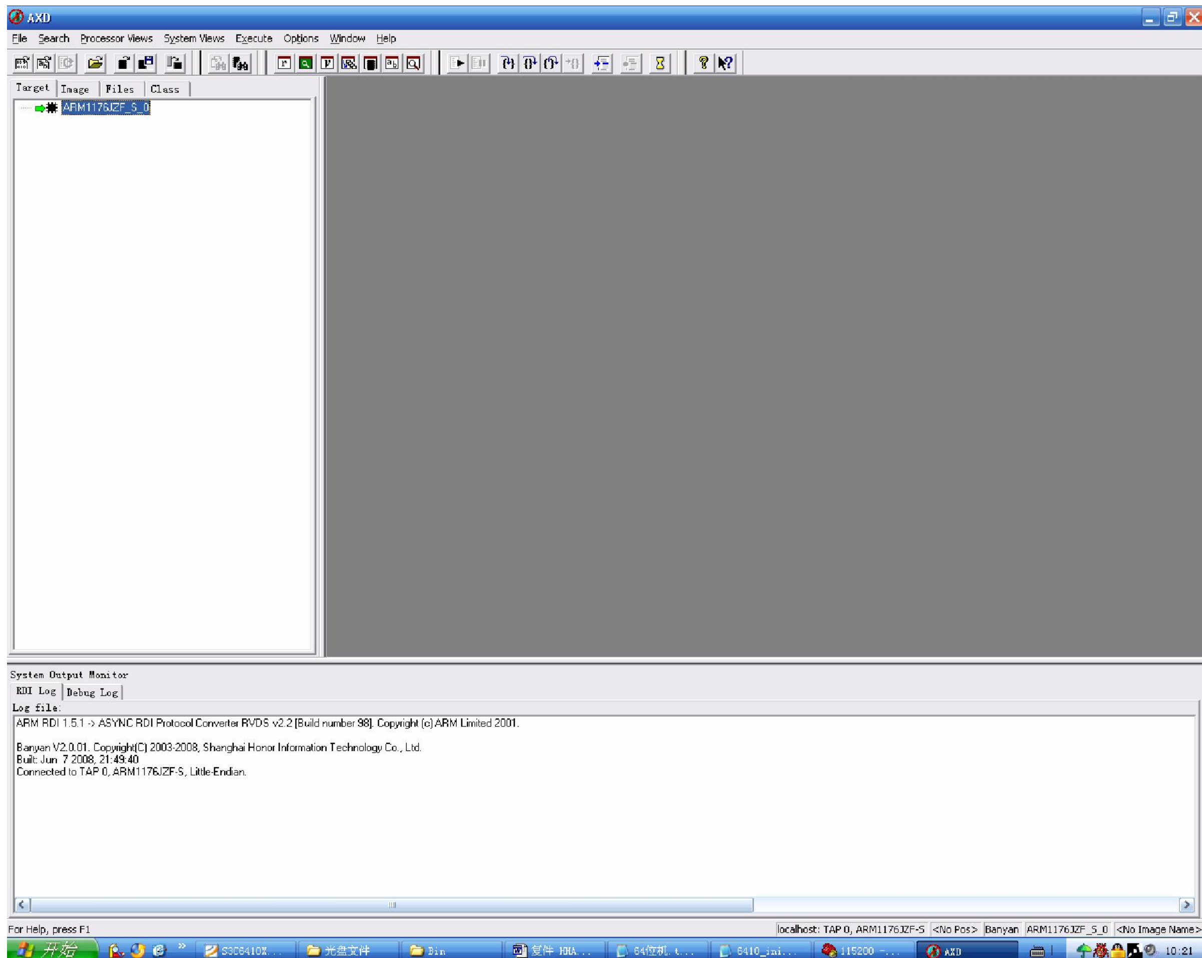
打开菜单 "Option"，选择 "Configure Target··"



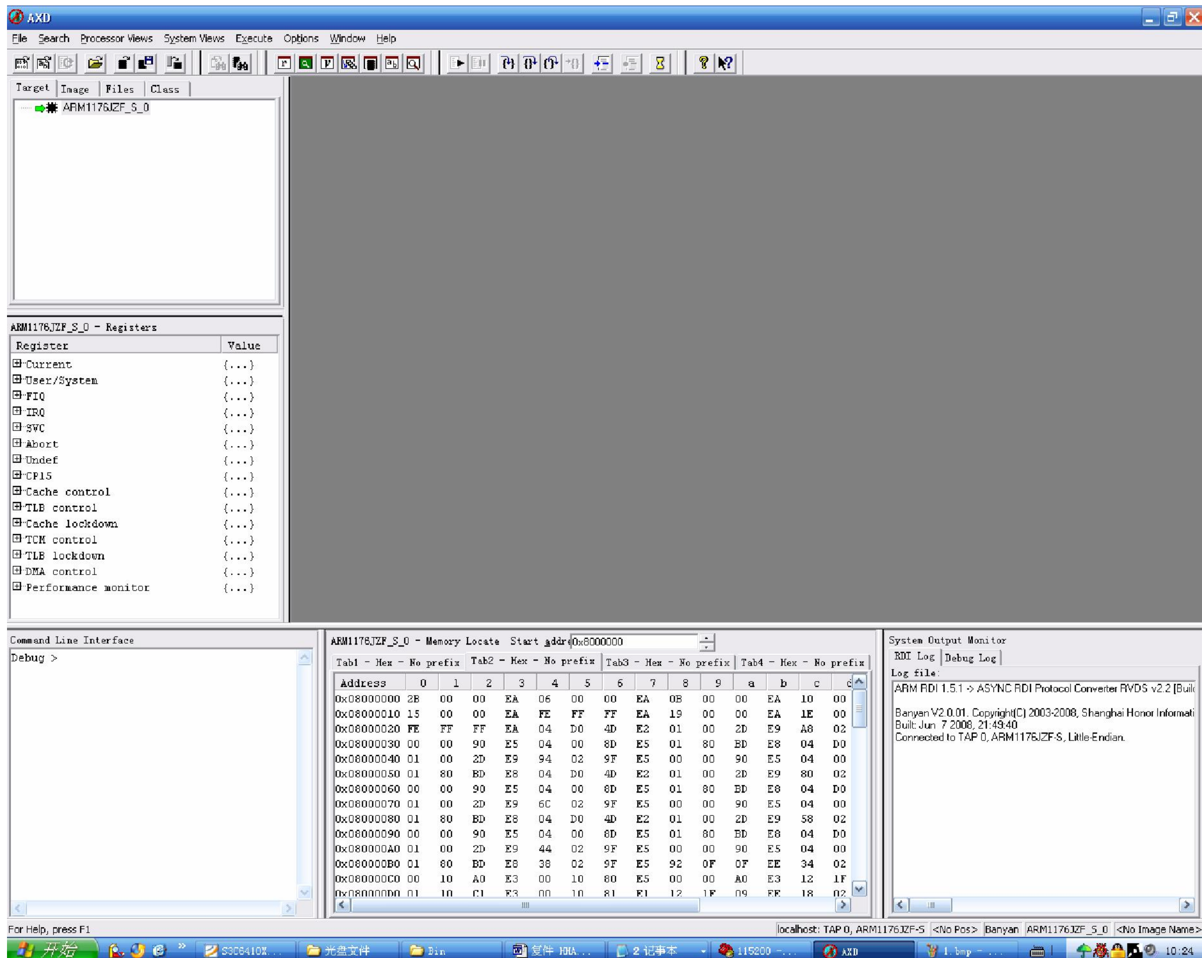
出现如下对话框：



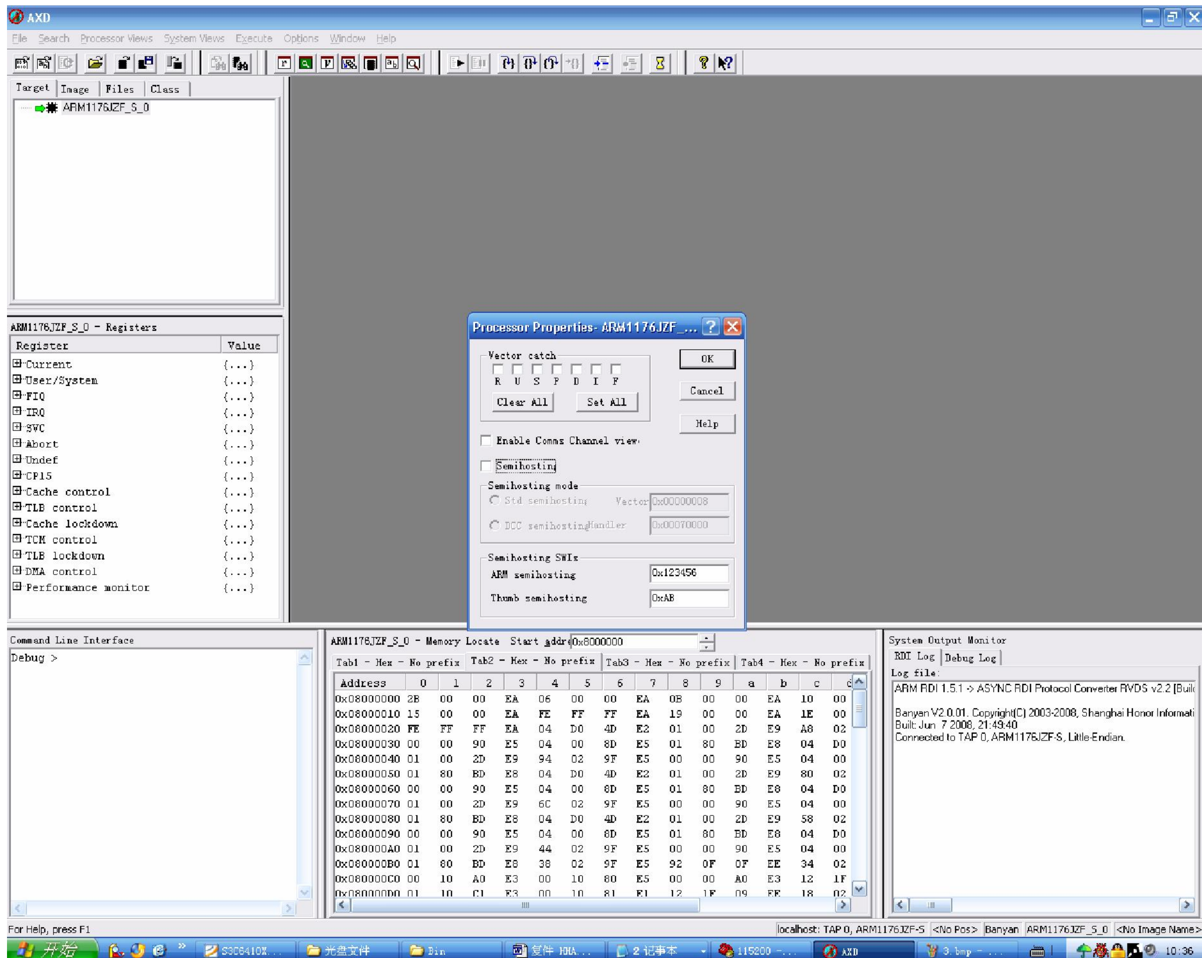
单击“Add”命令按钮，选择“Banyan.dll”（这个文件一般情况下放在仿真器驱动安装目录下），单击“OK”。



表明已经与仿真器建立了连接。单击工具栏上的 Command Line（alt+l），ctl+l命令按键，会出现Command Line Interface命令窗格和 memory窗口。


AXD
File Search Processor Views System Views Execute Options Window Help
Target Image Files Class
ARM1176JZF_S_0
ARM1176JZF_S_0 - Registers
Register Value
Current
User/System
FIQ
IRQ
SVC
Abort
Undef
CP15
Cache control
TLB control
Cache lockdown
TCM control
TLB lockdown
DMA control
Performance monitor
Command Line Interface
Debug >
ARM1176JZF_S_0 - Memory Locate Start addr 0x8000000
Tab1 - Hex - No prefix Tab2 - Hex - No prefix Tab3 - Hex - No prefix Tab4 - Hex - No prefix
System Output Monitor
RDI Log Debug Log
Log file:
ARM RDI 1.5.1 -> ASYNC RDI Protocol Converter RVDS v2.2
Banyan V2.0.01. Copyright(C) 2003-2008, Shanghai Honor Informati
Built: Jun 7 2008, 21:49:40
Connected to TAP 0, ARM1176JZF-S, Little-Endian.
For Help, press F1
localhost: TAP 0, ARM1176JZF-S <No Pos> Banyan ARM1176JZF_S_0 <No Image Name>

C清除 CPU状态



说明：另外可以用华恒的 HH ICE仿真器烧写引导代码，具体说明可参考光盘解压目录下的 HH ICE目录下的文档。

附录 G ubuntu 8.04 中若干服务的配置与安装

一. 省去每次都加 sudo:

sudo -s -H，输入第一个用户的密码，以后在此终端下执行命令时，就不用在每个命令前面加上 sudo 了。

二 .tftp,tftpd

需要在 PC机的根目录下建立 tftpboot目录并赋于此目录相关权限：

sudo mkdir /tftpboot ；sudo chmod 777 /tftpboot

ubuntu 8.04配置如下：

直接安装以下三个包即可。

sudo apt-get install tftp atftpd xinetd

8.04的 tftp服务下载文件时的默认目录 (/tftpboot)是在 /etc/inetd.conf中定义的。可以更改此目录中的相关设置，更改后使用 sudo /etc/init.d/xinetd restart重新启动一下服务即可。

ubuntu 7.04或 7.10配置除了按照上面的安装外，还需要作如下修改：

把 /etc/init.d/atftpd 文件中的 if ["$USE_INETD"="true"]中的 true修改成 faulse, 若要使用开发板从 PC机的 /tftpboot目录下载文件，还需要在 /etc/xinetd.d目录下存放有 tftp这样一个文件，如果没有，就手动的建立一下，文件内容如下：

```
service tftp
{
    disable         =no
    socket_type     =dgram
    protocol        =dup
    wait            =yes
    user            =root
    server          =/usr/sbin/in.tftpd
    server_args     =-s /tftpboot
}
```

三 nfs配置

ubuntu 7.04中，可以直接使用“系统”->“系统管理”->“共享文件夹”来设置 NFS或 samba 共享。



ubuntu 8.04系统中，没有“共享的文件夹”菜单，需要手动安装。

sudo apt-get install nfs-kernel-server

sudo vi /etc/exports

输入 /share *(rw,sync)

sudo /etc/init.d/nfs-kernel-server restart，在 shell提示符下键入 exprotfs 如果看到“/share <world>”或者使用 showmount -e如果看到

Exprot list forwk-desktop:

/share *

则说明安装配置成功。

四．无法运行 makemenuconfig

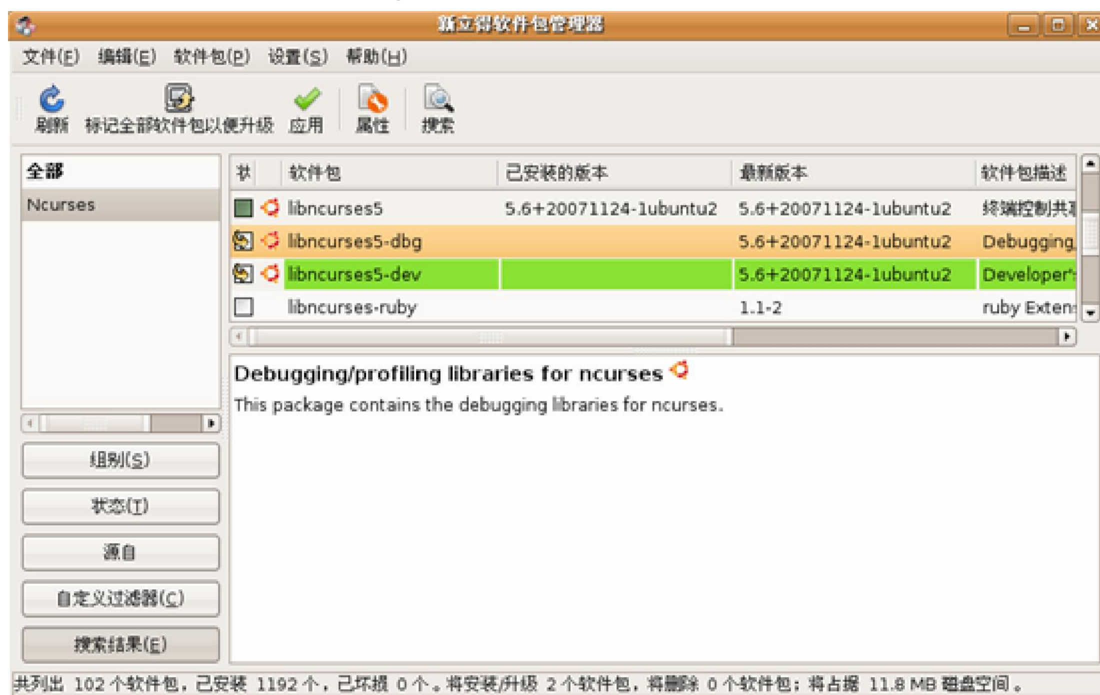
刚安装完后的 ubuntu系统 执行 sudomakemenuconfig时会有以下提示：

…… ……

```
make[1]:Entering directory `/HHARM2410/kernel/scripts/lxdialog'
/usr/bin/ld:crt1.o:No such file:No such file or directory
collect2:ld 返回 1
-e
>> Unable to find the Ncurses libraries.
>>
>> You must have Ncurses installed in order
>> to use 'make menuconfig'

make[1]:*** [ncurses] 错误 1
make[1]:Leaving directory `/HHARM2410/kernel/scripts/lxdialog'
make:*** [menuconfig] 错误 2
```

一看即知，上面缺少 Ncurses库 使用 sudo apt-get install libncurses5-dbg安装，或者使用“新立德软件包管理器”安装 libncurses5-dbg和 libncurses5-dev即可



五． Samba

sudo apt-get install samba

安装完以后，就可以在 windows的我的电脑的地址栏中输入此 linux 机器的 IP地址，然后就可以看到已经共享出来的资源了。也可以修改 /etc/samba/smb.conf 文件，重新设置相关共享资源，例如可以在 smb.conf文件的最后加入如下内容，

```
[Share]
    path = /share
    available = yes
    browseable = yes
    public = yes
    writable = yes
```

这样就把 /share目录下共享出去了，准许别人写此目录了。最后一步就是重启 samba服务。

sudo /etc/init.d/samba restart

现在网络上的计算机 (windows或 linux都可以通过 samba访问共享目录了。

如果是 ubuntu 7.04或 7.10版本，还需要使用“ sudo smbpasswd -a 用户名 ”设置一下密码，samba用户的密码可以和登录时不同。

六 .telnetd服务

8.04只需要执行 **sudo apt-get install telnetd**，然后 **sudo /etc/init.d/xinetd restart**，然后就可以通过网络使用 telnet登录此台 PC机了。

7.04或 7.10还需要如下配置

1. sudo apt-get install xinetd
2. sudo apt-get install telnetd
3. 修改或创建以下两个文件： /etc/xinetd.conf和 /etc/xinetd.d/telnet

第一个文件：内容如下

```
##--------------------------------------##
# 《/etc/xinetd.conf》
# Simple configuration file for xinetd
#
# Some defaults, and include /etc/xinetd.d/
defaults
        {
        instances = 60
        log_type = SYSLOG authpriv
        log_on_success = HOST PID
        log_on_failure = HOST
        cps = 25 30
        }
includedir /etc/xinetd.d
##--------------------------------------##
```

第二个文件：内容如下

```
##--------------------------------------##
# 《/etc/xinetd.d/telnet》
# default: on
# description: The telnet server serves telnet sessions; it uses \
# unencrypted username/password pairs for authentication.
service telnet
        {
       disable = no
flags = REUSE
socket_type = stream
wait = no
user = root
server = /usr/sbin/in.telnetd
log_on_failure += USERID
        }
##--------------------------------------##
```

4.重启 xinetd: **sudo /etc/init.d/xinetd restart**

七：配置 ftp，客户端可以安装 ncftp，服务端可以安装 vsftpd和 proftpd

1.vsftpd

安装后，只能匿名登录，并且无法写，需要修改 /etc/vsftpd.conf中，把 local_enable=Yes前面的#去掉就允许本地用户名登录了，把 write_enable=YES中的 #号去掉，就可以允许用户写了，别忘了，修改完后，重新启动一下 vsftpd， sudo /etc/init.d/vsftpd restart

默认 ftp都是登录到用户自己的主目录，也可以在 vsftpd.conf中设置 local_root参数设置登录目录，例如在 vsftpd.conf 中的最后设置 local_root=/backup/share，则不管谁登录，都登录到

/backup/share目录。

2.proftpd，装完过后，修改 /etc/proftpd/proftpd.conf

```
ServerName"Debian"
ServerTypestandalone
DeferWelcomeoff

MultilineRFC2228on
DefaultServeron
ShowSymlinkson

TimeoutNoTransfer600
TimeoutStalled600
TimeoutIdle1200

DisplayLogin                welcomemsg
DisplayFirstChdir           message
ListOptions             "-l"

DenyFilter\*.*/
```

加上以下这一句就行了。

DefaultRoot /share/　　/share是 linux要共享的目录名，如果不改，ftp客户端登录时使用什么用户就会进入此用户的主目录。

八． Mount命令挂载 PC机的共享目录后，无法执行交叉编译生成的文件

可以在 mount中加入几个参数，命令格式如下：

mount -o rsize=1024,wsize=1024 -o nolock 192.168.2.111:/backup /mnt

九． PDF格式文件乱码：

如果使用 Evince文件查看器看 PDF格式文件乱码，可以安装简体中文版的 xpdf，

sudo apt-get install xpdf-chinese-simplified xpdf-chinese-traditional poppler-data

十．不同的 shell引起的无法执行 shell脚本

有时在 Redhat9.0上可以执行的脚本，在 ubuntu上面执行时报错，必须用

“ source 脚本文件 ”

的形式执行，甚至有时这样也无法执行，那可能是 ubuntu的 /bin/sh的链接到 dash这个 shell中了，不同的 shell引起的问题，可以把 dash shell移除。

sudo apt-get remove dash

十一 修复 grub

如果一块硬盘安装的是双引导系统（一个 windows，一个 linux），当重新安装 windows时，如果 grub不在 MBR或已经格式过硬盘 MBR，grub可能会被冲掉，使用下面方法修复。

1从 LiveCD（如 Ubuntu Live, Knoppix, Mepis或者相类似的）启动。

2.打开终端。切换为超级用户（SuperUser）。（在 Ubuntu中输入 ” sudo -i”，在 Ubuntu以外的其它发行版中输入 ” su”）。输入根用户密码。如果是 Ubuntu的 LinveCD可能不需要密码，如果在图形界面下无法打开终端，可以按一下 Ctrl+Alt+Fn(n为 1,2,3...6)进入字符终端。

3输入 ” grub”，会出现 GRUB命令提示。

4输入 ” find /boot/grub/stage1″。会出现如同 ” (hd0)” 样的结果，在我的电脑上是 ” (hd0,3)”。在下面的命令中使用前面 find得到的结果。

5输入 ” root (hd0,3)”。

6输入 ” setup (hd0,3)”。这里是关键。在一些指南中使用 ” setup (hd0)” ,则是将 grup写入 MBR。如果您想将 GRUB写入您的 Linux根分区的话，则需要在逗号之后加上数字，形如 ” (hd0,3)”。 如果成功，会出现 “ Success..字样 ”

7输入 ” quit”。

8取出 LiveCD，重新启动电脑，看看 grub有没有恢复？

十二 双硬盘双引导

如果您的 PC 机上有两个硬盘，一个硬盘单独安装了 windows，另一块硬盘单独安装了 linux，则可以修改 grub让其既可以引导 linux，也可以 windows。需要在 BIOS中把安装有 linux的硬盘作为启动盘，通过 grub可以选择是进入 linux还是进入 windows。

假设在第二块硬盘的第三个分区上安装了 WindowsXP系统，可以修改 /boot/grub/menu.lst文件用以引导 widonwsXP，在 menu.lst文件中加入以下内容：

```
title      MicrosoftWindowsXP Professional
root (hd1,2)
savedefault
makeactive
map (hd0) (hd1)
map (hd1) (hd0)
chainloader  +1
```

【注意】！ root语句中的硬盘号是 Windows系统所在的硬盘号，不是“硬盘交换”后的第一块硬盘。

十三 其它。

还有一些其它的需要安装的一些软件，例如，像 ubuntu默认安装的是简易的 vim-tiny，您可以安装 vim，如果您英文不太好，可以安装星际译王等等，其它更详细的的安装及帮助请以下

IRC:#ubuntu-zh on irc.freenode.net

邮件列表 :ubuntu-zh

主站点 :ubuntu.org.cn

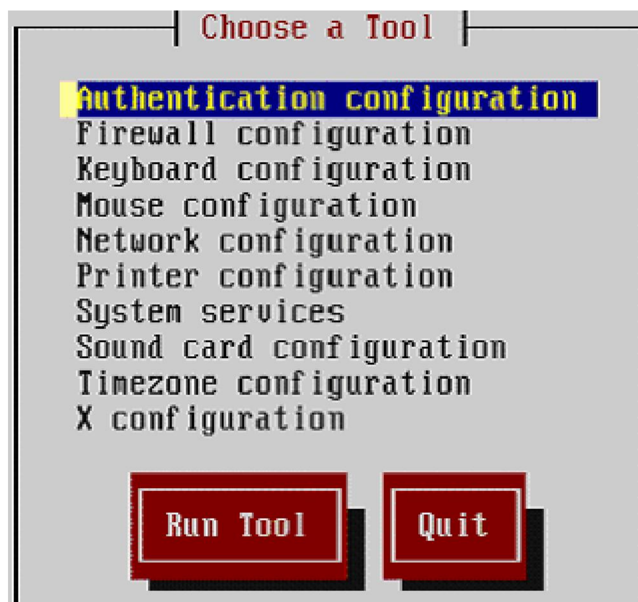
论坛 : forum.ubuntu.org.cn

Wiki文档 :wiki.ubuntu.org.cn

附录 H Redhat 9.0 NFS 和 TFTP 服务器的配置

1.NFS的配置：

首先在 REDHAT LINUX PC机上 shell提示符 [root@…]#执行 setup 弹出菜单界面后：



选中 System services 回车进入系统服务选项菜单，在其中选中 [*]nfs，然后退出 setup界面返回到命令提示符下。下面设置一下 nfs共享目录：

vim /etc/exports

将这个默认的空文件修改为只有如下一行内容，

/ (rw) /即把根目录共享成可读可写，rw,sync,no_root_squash 表示访问限制，/ 和 (rw)之间要要留空格

然后保存退出（ wq），然后执行如下命令：

运行以下命令启动 NFS服务。

/etc/rc.d/init.d/nfs restart 或 service nfs restart

```
Shutting down NFS mountd:        [ OK ]
Shutting down NFS daemon:        [ OK ]
Shutting down NFS quotas:        [ OK ]
Shutting down NFS services:      [ OK ]
Starting NFS services:           [ OK ]
Starting NFS quotas:             [ OK ]
Starting NFS daemon:             [ OK ]
Starting NFS mountd:             [ OK ]
```

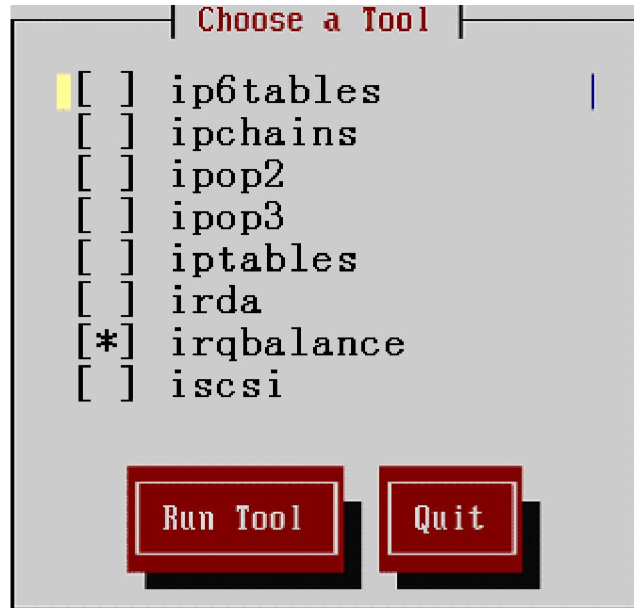
这样 NFS的配置就完成了！启动完成后，可用如下办法简单测试一下 NFS是否配置成功——PC机自己 mount自身。例如在 PC机的根目录下执行：(假定 PC机的 IP是 192.168.0.8)

mount 192.168.0.8:/ /mnt

然后到 /mnt目录下使用 ls命令看看是否可以列出 IP为 192.168.0.8这台机器根目录（/）下的所有文件和目录，如果可以列出目录，则说明 mount成功，NFS配置成功。

2.TFTP服务的配置：

在 PC机上执行 setup, 选择 System services, 将其中的 tftp一项选中（出现 [*]表示选中），并去掉 ipchains和 iptables两项服务（即去掉它们前面的*号）。



最后，退出 setup, 执行如下命令以启动 TFTP服务：

service xinetd restart

配置完成后，建议简单测试一下 TFTP服务器是否可用，即自己 tftp自身，例如在 PC机上执行：

cd /

cp /etc/inittab /tftpboot/ //随便拷贝一个文件到 /tftpboot目录下以供下面使用 tftp命令下载，下面以 etc/inittab文件为例进行说明。

tftp 192.168.0.8

tftp> **get inittab**

如果在 /tftpboot目录里没有下载的文件，会提示您没有找到相关文件。若出现如下信息：

Received 741512 bytes in 0.7 seconds

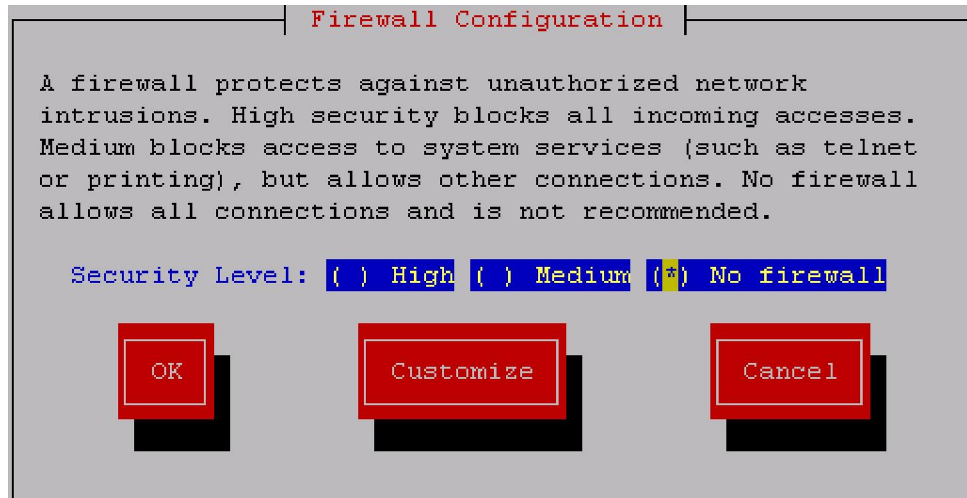
就表示 TFTP服务器配置成功了。在根目录下就会存在刚才下载的 inittab文件了；若弹出信息说：Timed out, 则表明未成功，或者用如下命令查看 tftp服务是否开通：

netstat -a|grep tftp

若 TFTP服务器没有配置成功，需要按照上述步骤重新检查一遍。

(3) 关闭防火墙：

键入 **setup**，选择 Firewall configuration, 选中 No firewall关闭防火墙。



『说明』

1. REDHAT LINUX PC自己 mount自己成功也不能完全说明 NFS就可以工作了，因为还有一个防火墙的问题，一般的我们建议客户在安装 REDHAT LINUX时就默认选择 NO FIREWALL，但若客户没有这样选择的话，REDHAT 启动时会加载防火墙规则，这样你自己能 mount自己，但其它 PC和开发板也无法 mount这台 PC。

【注意】！ setup里面的防火墙显示永远都是 HIGH，这个是 REDHAT一直的一个小 BUG，即使你安装时默认选择了 NO FIREWALL，setup里面也照样会显示防火墙设置是 HIGH的，这个可以不必理会。只要你选择了一次 NO FIREWALL就可以了。另外，建议对 LINUX操作不是非常熟悉的客户，务必请阅读我们手册附录的" LINUX常用命令"。

2. 有时若根目录不让读写，在 PC机的命令提示符下执行 chmod 777 /试试 。

3. REDHAT在加入网关设置后，网络建立 TCP链接非常慢，例如 FTP/TELNET/NFS等都是，建立链接后，以太网通信还是很快的。若您安装网络时加入了网关，就会出现你在开发板 NFSmount REDHAT PC时，敲入 mount -o nolock XXX XXX XXX XXX :/ /mnt后非常长的时间不返回 SHELL提示符，就一直停在那里，这就是建立连接的时间非常的长。一般的，遇到这种情况，建议在 REDHAT LINUX PC上执行

route del default

即去掉网关，然后再进行 NFSmount等操作就会非常快了。

4. 测试 NFS你必须用网线将开发板和 REDHAT LINUX PC连接起来才行，连接的方式有两种：一是开发板和 PC都用普通的网线接到 HUB或者交换机上；二是用交叉网线将开发板和 PC直连起来，注意！这种网线是特制的，内部收发交换的！（一头按普通网线的做法，另一头把 1,3;2,6交换一下），测试开发板和 PC网络是否连通的方式是：开发板正常启动 LINUX后，在 minicom里面 ping一下 PC看是否能通即可，当然了，用 PC来 ping开发板看是否通也是可以的。 注意：开发板必须启动到 LINUX后才能 ping通，开发板处在 bootloader阶段一般是无法 ping通的，即使这时开发板的 bootloader初始化了以太网也不行。一般情况下，如果不设置网关，使用 ping命令时，需要开发板和 PC机在同一个网段。

若开发板跟 PC的网络不通，mount报错信息为：

```
#mount -o nolock 192.168.0.8:/     /mnt
mount: RPC: Unable to receive; eNmo = No route Fo host
mount: Sfsmount failed: Bad file descriptor
mount program didn't pass remote address!
```

mount:Mounting 192.168.0.8:/ on /mnt failed: Invalid argument

6. 关于 tftp 服务安装的问题：有些客户在安装 REDHAT LINUX 时，没有选 Custom-->Packages里面选择 everything, 导致没有安装 tftp服务（典型的就是安装完成后根目录下没有 tftpboot目录，setup-->system services里面也没有 tftp选项），这时若要重新完全安装就太麻烦了，可按以下单独安装该服务试试：

安装 tftp 的方法：

A首先找到 tftp-server-0.32-4.i386.rpm 或 foo-1.0-2.i386.rpm包 (在 Redhat9的第 3张光盘里面）

rpm -ivh tftp-server-0.32-4.i386.rpm

B在根目录下建立 tftpboot目录。

C. 修改 /etc/xinetc/tftp为以下内容 (修改前最好备份一下）, 没有就建立

```
service tftp
{
disable = no
socket_type          = dgram
protocol             = udp
wait                 = yes
user                 = root
server               = /usr/sbin/in.tftpd
server_args          = -s /tftpboot
per_source           = 11
cps                  = 100 2
flags                = IPv4
}
```

C重启 tftp服务。

/etc/init.d/xinetd restart

7 NFS的服务的配置

如果没有完全安装，相应的 NFS也可能没有安装，那就试试下面的步骤。

A . rpm -q nfs-utils 如果没有安装 从安装光盘上找到 nfs-utils 的安装包，

rpm -ivh nfs-utils-xxx.rpm (xxx为您使用的 nfs的版本）

B. 设置 NFS共享的目录，请看前面的章节。

C. 重启 nfs服务，请见前面章节。

『说明』

如果没有配置好上面的 TFTP服务，开发板就没有办法使用 tftp通过以太网下载映像文件。

如果 NFS没有配置好，则应用程序的开发就没有办法使用 NFS服务，这一项不是必须的，但有这一项服务，可以大大方便调试应用程序的开发。

一般情况下，就用宿主机做 TFTP 服务器，即用同一台 Linux PC既做宿主机，也在此台 PC机上配置其它的服务像 NFS, 让其变成 NFS服务器，当然也可以使用其它机器配置以上服务。

附录 I 参考资料

A） S3C6410X_UM_PreliminaryRev0.2_20071019.pdf
B）《UNIX 环境高级编程》，W.Richard Stevens 著，机械工业出版社；
C）《Linux 设备驱动程序》，ALESSANDRO RUBINI 著， LISOLEG 译，
中国电力出版社；
D）《嵌入式 LINUX 设计与应用》（清华大学出版社 2002 年出版）。

售后服务与技术支持

在使用华恒嵌入式 Linux 开发套件进行开发时所遇到的一切问题，均可在华恒嵌入式 Linux 技术论坛：http://bbs.hhcn.com/站点上得到专家们快捷、准确的回答。

另外，还可以发 mail 到华恒公司工程师组电子信箱 hharm-support@hhcn.com，或者直接致电华恒公司，您都会得到及时的答复。

http://www.hhcn.org 上还提供支持华恒嵌入式 Linux 开发板应用开发的技术资料、源代码、开发源代码的 GNU 项目以及开发套件软件的升级版本和补丁，使得基于华恒嵌入式 Linux 开发板进行产品开发更快捷、便利。还有，技术支持论坛上提供华恒科技各种系列开发平台的"常见问题解答"的链接。

http://www.hhcn.com/chinese/hharmfaq.html ：华恒 ARM 产品 FAQ
http://www.hhcn.com/chinese/hhco5272r1faq.html ：华恒 Coldfire产品 FAQ
http://www.hhcn.com/chinese/hhppcfaq.html ：华恒 PowerPC产品 FAQ
http://www.hhcn.com/chinese/hhdrez328_faq.html ：华恒 DragonBall产品 FAQ
http://www.hhcn.com/chinese/embedlinux-res.html ：嵌入式 Linux开发资源

华恒嵌入式 Linux 开发套件整体保修期为六个月，第一个月可免费更换，但 CPU烧坏、flash 烧毁、SDRAM、LCD屏电源、插座、JTAG卡及电缆等易耗件及人为损坏不在保修范围之内。

如果您在使用华恒产品时遇到故障，请首先与华恒科技的售后技术支持联系，需要返回华恒公司维修的，请认真填写下面表格中您的联系方式（或夹上您的名片）、返修清单和故障现像，把此表格打印出来放到包裹里。因为您寄过来的包裹到华恒公司时，写在包裹外面的您的地址可能已经看不清楚了，或您的地址已经变更，谢谢合作。

华恒科技的联系方式请见本手册的最后一页，并与售后技术支持人员联系确认"收信人"。

华恒科技会在收到返修器件的三个工作日内给出初步结果，特殊情况（像 CPU等重要器件烧坏）一周时间给出初步结果。

警告！

请务必注意静电的防护。超过任何最大承受值，均会对产品产生永久损害。同时，不推荐在临界状态使用产品。

客 户 返 修 单

编号：____________

<table>
<tr><td rowspan="6">华恒客户填</td><td>单位名称</td><td colspan="3"></td></tr>
<tr><td>单位地址</td><td colspan="3"></td></tr>
<tr><td>联 系 人</td><td></td><td>联系电话</td><td></td></tr>
<tr><td>邮　　编</td><td></td><td>Email</td><td></td></tr>
<tr><td>返修事由、故障现像</td><td colspan="3"></td></tr>
<tr><td>返修器件清单</td><td colspan="3">1.
2.
3.
4.
5.</td></tr>
<tr><td rowspan="4">华恒公司填</td><td>我方实测故障</td><td colspan="3"></td></tr>
<tr><td>故障原因</td><td colspan="3"></td></tr>
<tr><td>解决办法</td><td colspan="3"></td></tr>
<tr><td>维修人员签名</td><td></td><td>维修部主管签名</td><td></td></tr>
<tr><td></td><td>备注</td><td colspan="3"></td></tr>
</table>

公司简介：

华恒科技是获得高新技术企业认证及双软企业认证的嵌入式 Linux开发平台及 OEM 板级硬件提供商，是中国软件行业协会嵌入式分会理事单位、安徽省软件行业协会理事单位会员、安徽省产业化项目重点支持企业、安徽省火炬计划项目企业。公司成立于 1998年，同步于全球嵌入式 Linux 产业发展的萌芽期，是国内最早进行嵌入式 Linux软硬件研究与开发的企业。

从全球嵌入式 Linux行业的起步时期，华恒科技开始嵌入式 Linux的软硬件技术研究。华恒科技自主研发的历史见证了国内嵌入式技术的发展和深化 -华恒科技不断支持最新的嵌入式处理器技术并不断为各种行业引进嵌入式 Linux技术。目前华恒科技已经成为国内自主研发产品种类最丰富、解决方案种类最丰富，具有强大行业影响的嵌入式技术服务提供商。

华恒科技在发展嵌入式 Linux技术的过程中，不断地与全球顶级的半导体公司开展深入的技术合作，整合丰富的技术资源为客户提供服务。华恒科技在嵌入式 Linux以及 uCLinux方面强大的技术能力，以及帮助客户成功设计产品、实现量产的软硬件技术经验，为华恒科技赢得了与半导体厂商长期的合作关系。

华恒科技与上游半导体厂商进行深入技术合作，数年来完成的大量具有技术先导性的嵌入式项目。包括与飞思卡尔半导体 NCSG（网络与计算机系统事业群）合作研发的基于 PowerPC处理器的防火墙主板、网关、高性能嵌入式计算机设计方案；与飞思卡尔半导体 TSPG（交通标准产品事业群）合作研发的基于 Coldfire处理器的机顶盒、 POS机、家庭媒体中心设计方案；与飞思卡尔半导体 WMSG（无线与移动系统事业群）合作研发的基于 iMX 处理器的 MP4播放器方案；与 ADI公司合作，完成基于 Blackfin处理器的 uCLinux操作系统的移植、基于 Blackfin处理器的 MP4播放器方案、视频监控方案设计等。

目前，华恒科技是飞思卡尔半导体（原摩托罗拉半导体）的全球设计联盟成员、ADI的 DSP 技术合作伙伴以及 ARM 公司 ARM Connected Community成员伙伴。华恒科技与其他国际知名的半导体公司如 Intel 三星、威盛、华邦等也保持商务与技术方面的联系与合作。

公司主页：

http://www.hhcn.com 华恒科技产品网站
http://www.hhcn.org 华恒科技技术资料网站

公司论坛：

http://bbs.hhcn.com

技术支持邮箱：

hhbf-support@hhcn.com Blackfin系列产品技术支持邮箱
hhcf-support@hhcn.com Coldfire 系列产品技术支持邮箱
hharm-support@hhcn.com ARM 系列产品技术支持邮箱
hhppc-support@hhcn.com PowerPC系列产品技术支持邮箱
hhmips-support@hhcn.com MIPS系列产品技术支持邮箱

总部地址：合肥市高新区梦园路 9号研发中心综合楼 3层 华恒科技

总部电话：**0551-5325652 / 5325653 / 5325631 / 5325173 / 5333155 / 5333156 / 5333157**
总部传真：**0551-5325323** 邮 编： 230088 电子邮件：market@hhcn.com